Tvorba softwarového projektu - metodická pomůcka pro základní a střední školy.

Creation of a Software Project – A Methodology Tool for Primary and Secondary Schools.

Bc. Marek Zmeškal

Diplomová práce
2012

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2011/2012

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Marek ZMEŠKAL**
Osobní číslo:	**A10368**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Učitelství informatiky pro střední školy**
Téma práce:	**Tvorba softwarového projektu – metodická pomůcka pro základní a střední školy.**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na téma projektové řízení tvorby softwarové aplikace.**
2. **Analyzujte aktuální možnosti a realizace softwarových projektů na základních a středních školách.**
3. **Navrhněte metodický postup pro tvorbu softwarové aplikace na základě zásad projektového řízení.**
4. **Realizujete aplikaci s detailním metodickým postupem respektujícím zásady projektového řízení.**
5. **Vyhodnoťte zvolené řešení v konfrontaci s praxí na školách.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **GUCKENHEIMER Sam, PEREZ Juan J.. Efektivní softwarové projekty. Vyd. 1. Překlad Jan Kuklínek, Jan Pokorný. Brno: Zoner Press, 2007, 255 s. ISBN 978-808-6815-626.**
2. **BECK Kent. Extrémní programování. Vyd. 1. Praha: Grada, 2002, 158 s. ISBN 80-247-0300-9.**
3. **ARLOW, Jim a Ila NEUSTADT. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. Vyd. 1. Překlad Bogdan Kiszka. Brno: Computer Press, 2007, 567 s. ISBN 978-802-5115-039.**
4. **GAMMA Erich, HELM Richard, JOHNSON Ralph, VLISSIDES John. Návrh programů pomocí vzorů: Stavební kameny objektově orientovaných programů. Praha: Grada Publishing, 2003. ISBN 80-247-0302-5.**
5. **MEŠKO Dušan, KATUŠČÁK Dušan. Akademická príručka. 1. slovenské vyd. Martin: Osveta, 2004, 316 s. ISBN 80-806-3150-6.**

Vedoucí diplomové práce: **doc. Mgr. Roman Jašek, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **24. února 2012**

Termín odevzdání diplomové práce: **21. května 2012**

Ve Zlíně dne 24. února 2012

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

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

ABSTRAKT

Cílem práce je vytvoření pomůcky, která bude v praxi využitelná učiteli při tvorbě softwarových produktů na základních a středních školách. Ta je vytvořena na základě výzkumu, který probíhal formou dotazníku. Vzniklá metodika je založena na rigorózním přístupu metodiky unifikovaného vývoje aplikací s prvky projektové výuky. Analýza metodik probíhala formou otevřených otázek, které měly tematický charakter.

Klíčová slova:

Metodika, projektová výuka, unifikovaný proces, tvorba aplikací, základní škola, střední škola.

ABSTRACT

The aim is to create tools that will be usable in practice, by teachers in developing software products in primary and secondary schools. It is created based on research that was conducted through a questionnaire. The resulting methodology is based on the rigorous methodology of a unified application development with elements of project education. The analysis methodology was conducted through open questions which had thematically character.

Keywords:

Methodology, design education, unified process, application development, elementary school, secondary school.

Rád bych touto cestou poděkoval mému vedoucímu diplomové práce doc. Mgr. Romanu Jaškovi, Ph.D. za cenné rady, připomínky a odborné vedení. Dále bych rád poděkoval všem, kteří mne během mého studia podporovali.

Non schoale sed vitae discimus.

-

Neučíme se pro školu, ale pro život.

Seneca

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské 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/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou 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/bakalářskou 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/bakalářské 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/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské 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 …….

Bc. Marek Zmeškal

OBSAH

ÚVOD

Téma diplomové práce jsem volil na základě svého zájmu o informatiku, zejména o programování. Po několika letech programování na střední škole a následně na vysoké škole mne oslovila práce v prostředí Visual Studio pro programovací jazyk C#. Po absolvování několika kurzů jsem postupně poznával rozsáhlost, kterou programátorský svět disponuje. Vzhledem k počátkům jsem zjistil, že ne všechny věci jsem dělal správně a bylo nutné tyto návyky upravit tak, aby odpovídaly kladeným požadavkům na zadaný softwarový projekt. Z tohoto důvodu jsem se rozhodl ve své diplomové práci provést rešerši na téma tvorby softwarové aplikace. Snažím se vystihnout důležité věci, které je možné využít na základních a středních školách. Opírám se především o rámcové vzdělávací programy pro základní vzdělávání a pro gymnázia, jenž určují jasné požadavky na absolventy škol. Protože informatika obecně je velice rozsáhlý obor, ne na všech školách se vyučuje stejným způsobem, a tím pádem ne všude bude žák disponovat stejnými znalostmi. Z uvedených důvodu provedu na základních a středních školách výzkum dotazníkovou metodou, kde cílem bude zjištění, jakým způsobem žáci v hodinách pracují. S výhledem do budoucna bude zmíněn i nejbližší vývoj a očekávaní ve školství. Posléze se pokusím navrhnout metodický postup, řešený projektovou výukou, jenž se bude snažit využít zásady projektového řízení. Výsledkem bude metodický postup s projektovou tvorbou výsledné aplikace, který je možné využít na základních a středních školách. Samotný výtvor bude konzultován s učiteli provozujícími výuku informačních a komunikačních technologií.

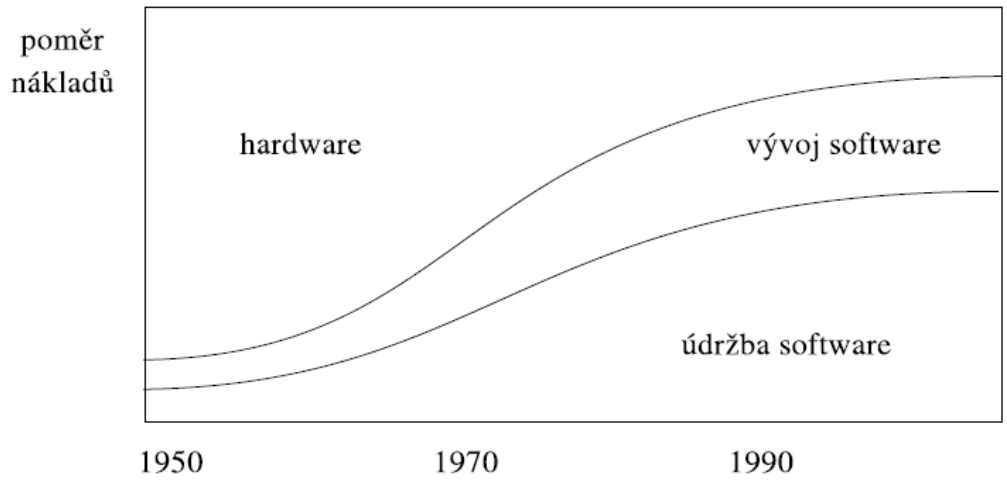
I. TEORETICKÁ ČÁST

1 SOFTWARE

Obecně existuje mnoho definic, co software je. Běžně se dá software charakterizovat jako programové vybavení počítače, někdy se nazývá i programovým systémem. Z hlediska uživatele je software možné kategorizovat jako aplikační programové vybavení umožňující provádět požadovanou činnost (např. zpracování textů, audia, videa, tvorba obrazových materiálů, atp.) a systémové programové vybavení, které umožňuje efektivně rozdělovat prostředky, jenž jsou požadovány uživatelem.[9][10] Z pohledu odborníků lze pojem software formulovat následně: *„Programovým systémem je chápán softwarový produkt, který je tvořen množinou programových jednotek (modulů, objektů, komponent, služeb) a jejich vzájemných vazeb"[8]*

1.1 Vývoj metodik tvorby softwaru

První programy pro počítače, které byly na bázi elektronek a spínačů relé, byly tvořeny vědci. Především proto, že se programovalo přímo ve zdrojovém kódu a znali podrobně hardware a dokázali tak vytvořit program přímo na míru pro tento stroj. První metodika, považující se za světovou, vznikla v letech 1968-1969, a to strukturované programování. Důležitou roli v tomto případě hrál fakt, že se výrazně začal měnit poměr mezi pořizovací cenou hardwaru, cenou softwaru a náklady vynaloženými na provoz (údržbu) softwaru – viz Obrázek 1.[9][10]



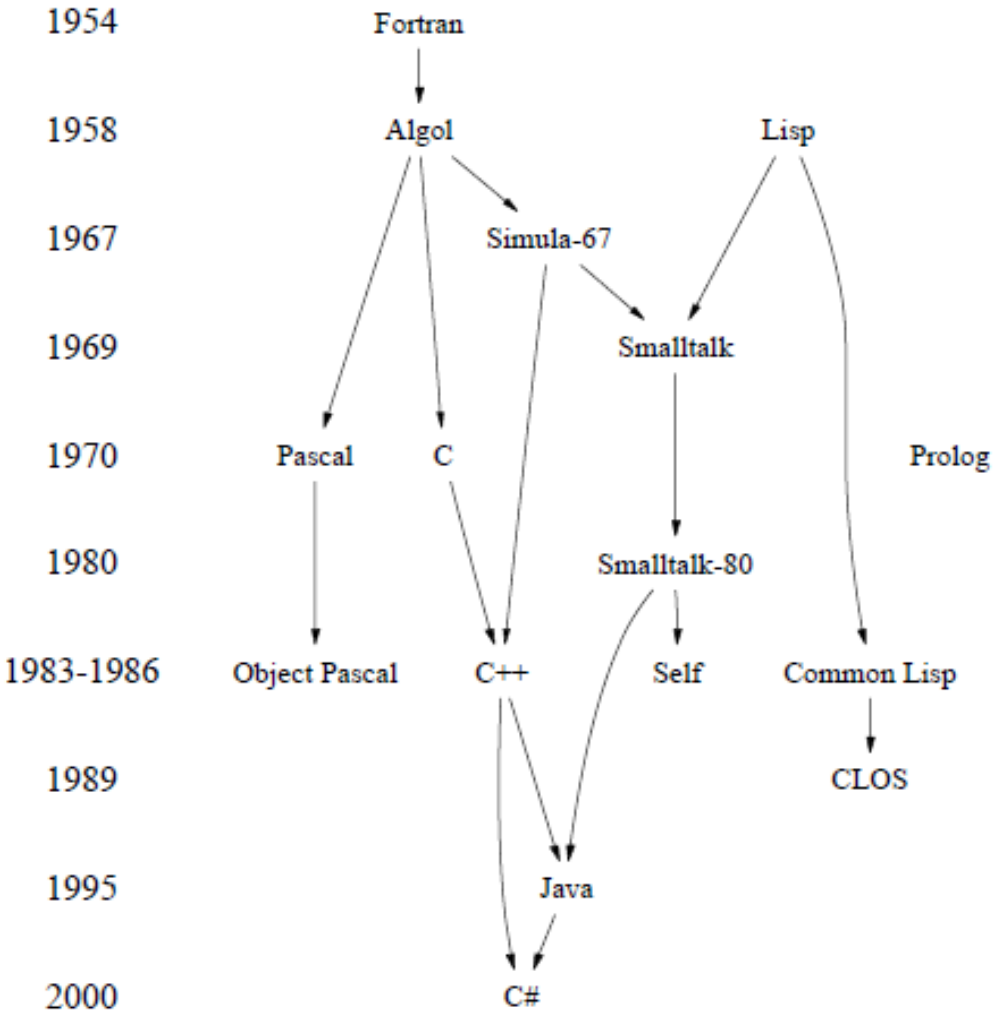
Obrázek 1 – Vývoj poměrů HW, SW a vývoj v letech 1950-2010. [10]

1.1.1 První generace 1945-1955

Téměř žádná metodika v této době neexistovala, byl pouze jeden tým, který se staral o běh počítače. V týmu byl programátor, konstruktér, operátor i technik. Úspěchem bylo ukončení výpočtu bez chyby. Programovalo se přímo ve strojovém jazyce.[9][10]

1.1.2 Druhá generace 1955-1965

Zlepšení HW části počítačů, zvyšují se nároky, začíná první obchod s počítači, nicméně strojový čas je velice drahý. Programování se začíná vyvíjet a programy se píší v jazyku symbolických adres – Fortran, Cobol, Algol. Začíná vývoj programovacích jazyků – Obrázek 2.[9][10]



Obrázek 2 - Vývoj programovacích jazyků. [10]

1.1.3 Třetí generace 1965-1980

Založeno na rozvoji HW, použití integrovaných obvodů. Zvýšena spolehlivost počítačů, ale pořád velké vstupně/výstupní prodlevy. Počítačový čas je drahý. V 60. letech vznik nových pojmů jako je softwarové inženýrství, softwarová krize. Problémy a jejich řešení započalo na konferencích v letech 1968-1969, jedná se o prodlužování a prodražování projektů, nízkou kvalitou výsledných softwarových produktů, problematikou s inovacemi a údržbou, špatnou produktivitou programátorů a výsledkem je řada neúspěšně ukončených projektů. Řešení pana Dijkstry vedlo ke strukturovanému programování. V 70. letech se začal klást důraz i na lidský faktor při programování, postupné zjemňování programů v podobě modulárního programování. Zavedl se abstraktní datový typ, snaha programovat v týmech, kde byl určen jeden vedoucí programátor. Co se týká přímo metodologie, byla zavedena strukturovaná analýza návrhu a začaly se využívat datově a procesně orientované metody, k tomu se přidalo vnímání životního cyklu tvorby SW, podpořen nárůst důrazu na etapy specifikace a návrhu. Byl to počátek vývoje procedur zaměřených na systematické testování.[9][10]

1.1.4 Čtvrtá generace od roku 1980

V 80. letech 20. století se začalo s větším používáním programovacích prostředí, navíc se začal vyvíjet SW na podporu metod programování. Toto období je charakterizováno nástupem mikroprocesorů a rozšířením ze sálových počítačů (jeden počítač pro celou firmu) na pracovní stanice (část zaměstnanců má přístup k PC), tím dochází ke „zlevnění“ počítačového času. Prosazování především GUI rozhraní a přechod na distribuované systémy s velkým rozvojem síťových aplikací. Počátek a vývoj objektově orientovaného programování, první pokroky v oblasti paralelního programování a počátek prvních návrhových vzorů. Navýšení důrazu na etapy nasazení a údržby softwaru pro údržbu verzí SW a řízení konfigurací.[9][10][11]

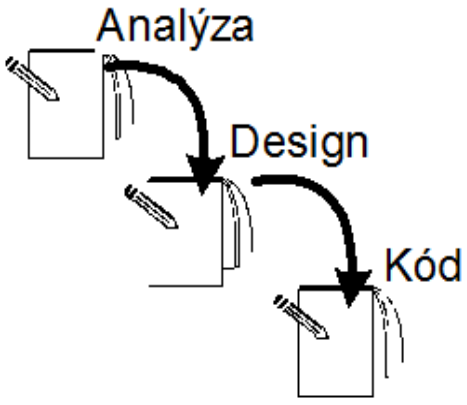
1.1.5 Poslední vývoj od 90. let a současnost

Na počátku 90. let představeny první návrhové vzory, které měly pomoci začínajícím programátorům s objektově orientovaným návrhem a dalšími specifikacemi. Vývoj SW je založený na znovu použitelnosti a komponentech. Hlavní myšlenku návrhovým vzorům dali Erich Gamma, Richard Helm, Ralph Johnson a John Vlissides, říkající si Gang of Four – GoF. Později (v roce 1993) založena nevýdělečná organizace The Hillside Group, mající

za cíl především zlepšení vývoje SW systémů. Po řadě metodik, které každá kladla důraz na různou část a chování projektů, vznikla metodika Unified Modeling Language – UML (v roce 1995). Následně v roce 1997 přijato skupinou Object Management Group – OMG, která funguje od roku 1989 a sdružuje podniky zabývající se tvorbou, vývojem a distribucí SW i HW a zabývá se standardizačním procesem. Do tvorby SW se prosazuje další sledování kvality a vývojem open source softwaru. Rozvojem a využíváním internetu se zapojují a vznikají nové techniky spolupráce. Rozšíření distribuovaných systémů a jeho metodám, technikám jeho tvorby. V současnosti je trendem podpora, servis a údržba SW, kdy se v mnoha případech jedná o outsourcing.[9][10][11]

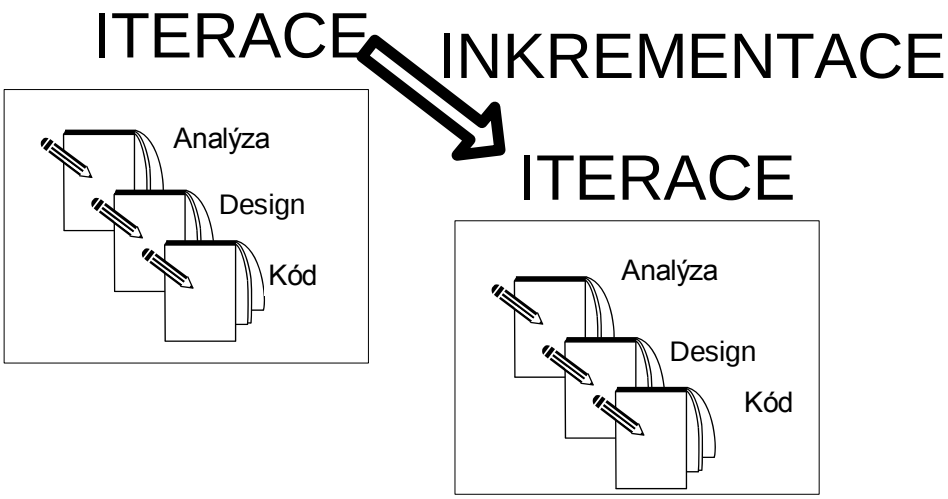
2 ŘÍZENÍ SOFTWAROVÝCH PROJEKTŮ

Každý softwarový projekt, na kterém pracuje více lidí nebo dokonce několik pracovních týmů, by měl obsahovat určitou metodiku práce, která dopředu koordinuje společnou činnost a uvádí další postupy při řešení problémových situací. Tvorba a řízení softwarových projektů je balancování na hranicích investic a rizik. Proto je důležité vhodné zvolení metodiky a její dodržování. Projekt obecně by měl splňovat několik hlavních a důležitých vlastností jako je dobrou funkčnost, vysokou kvalitu, nízké časové nároky na čas a vysokou hospodárnost se zdroji. Těchto pět základních požadavků na projekt spolu velmi úzce souvisí, zaměřením se na jednu část upravuje parametry dalších požadavků. Riziko v podobě nedokončení projektu se snižuje použitím metodik, nástrojů a dobrého vedení. Existuje několik metodik řízení projektů, kterými je možné se vydat. První z nich je tzv. „Tunel“, kde podstata spočívá v tom, že na počátku se vstupuje do neznáma, není zde žádná koncepce, vedení projektu je operativní. Vytváří se tímto způsobem tlak na realizaci projektu, tedy nalézt konec tunelu – projekt zdárně ukončit. Mezi další, pokročilejší, patří tzv. „Vodopád“ – Obrázek 3, hlavní myšlenkou je koncepce určující postupný vývoj. Čili nejdříve analytické dokumenty následně design dokumenty a na závěr dokumenty kódu, zdrojové dokumenty. Nevýhodu této metody je neobratnost a obtížné řízení lidských zdrojů.[1][3][8][11][12][13]



Obrázek 3 - Metoda vodopád. [12]

Protože se nejdříve provede kompletní analýza, následně až po ukončení přejdou pracovníci na design, po ukončení designových dokumentů tvoří výsledný zdrojový kód. Třetím vědeckým postupem je metoda inkrementální a iterativní (Obrázek 4). Její plné využití spočívá hlavně v objektově orientovaném programování (OOP), kdy je provedena analýza, design a kód v jedné malé iteraci.[10][12][13]



Obrázek 4 – Inkrementální a iterativní metoda. [12]

Rozšiřování systému v inkrementaci spočívá v přidávání funkcionalit objektů spolu s jejich novými, stále konzistentními, vnitřními stavy.[1][2][3][12][13]

2.1 Unifikovaný modelovací jazyk - UML

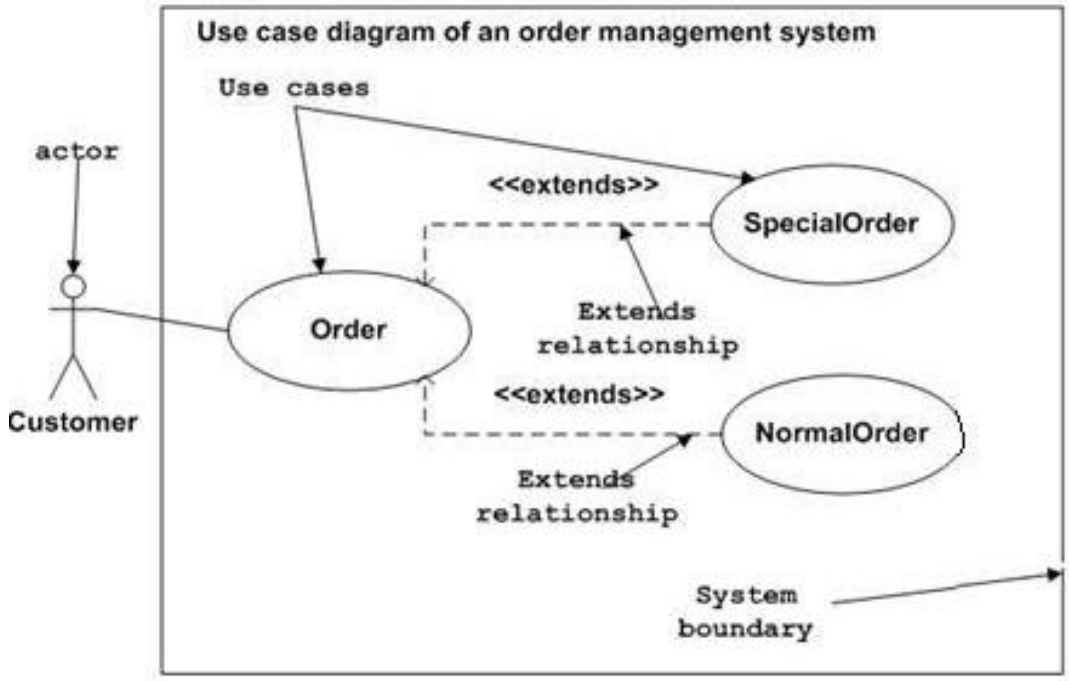
UML je standardem v objektově orientovaných analýzách a návrzích. I když samotný jazyk není metodikou, je pouze nástrojem pro tvorbu vizuálních syntaxí pro tvorbu, je nutné k tomuto návrhu použít metodiku. Standardem v programování je metodika Unifikovaného Procesu (UP). Modelovací jazyk utváří určitá vizuální hlediska na daný projekt z několika úhlů pohledů. V UML jsou tyto pohledy nazývané abstrakce. Rozděleny jsou do tří hlavních podle jejich úrovně. Na nejnižší úroveň abstrakce patří kódování. Na střední úrovni abstrakce je modelování návrhu a na nejvyšší úrovni abstrakce je analytické modelování. Cílem je kvalitní předávání informací v logických návaznostech, re-use agendy, transparentnost, malá chybovost, logické uspořádání komponent, dokumentace analýzy a návrhu, možnost předání práce dalšímu pracovníkovi, specializace pracovníka a tím umožnění jeho odborného růstu a zamezení nadměrného byrokratického omezování.[3][8][11][12]

2.1.1 Modely v UML

V UML je devět modelů. Které nahlíží na SW z hlediska vývojového procesu. Zabývají se především fází tvorby, analýzy a designu. Navíc je možná generace dokumentace, ovšem záleží na použitém nástroji.[3][11][12][13][14]

2.1.1.1 Use Case diagram

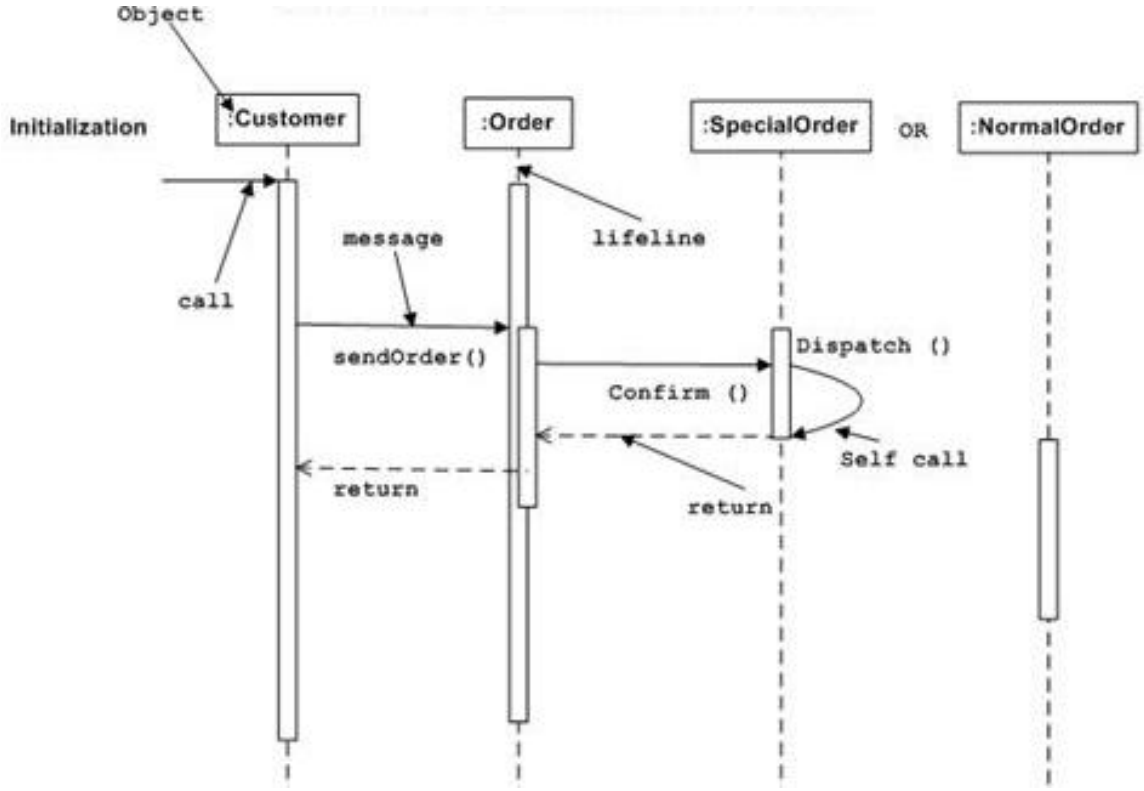
Pokud se jedná o překlad, užívá se nejčastěji o diagram případů užití. Popisuje se v něm hierarchicky seznam činností – Use Casů. Diagramy se tvoří na počátku. Jsou stavebním kamenem celého projektu. Většinou jsou tvořeny kompletně, postupem vývoje se již upravují výjimečně a minimálně. Na Obrázek 5 můžeme vidět ukázku případu užití. Máme nějakého uživatele – „Customer“, ten zadá do systému (znázorněného rámečkem ve tvaru obdélníku), nějakou objednávku – „Order“, tu je možné specifikovat dále jako speciální objednávku – „SpecialOrder“ a normální objednávku – „NormalOrder“.[3][11][12][14]



Obrázek 5 – Diagram Use Case - případu užití. [14]

2.1.1.2 Sequence diagram

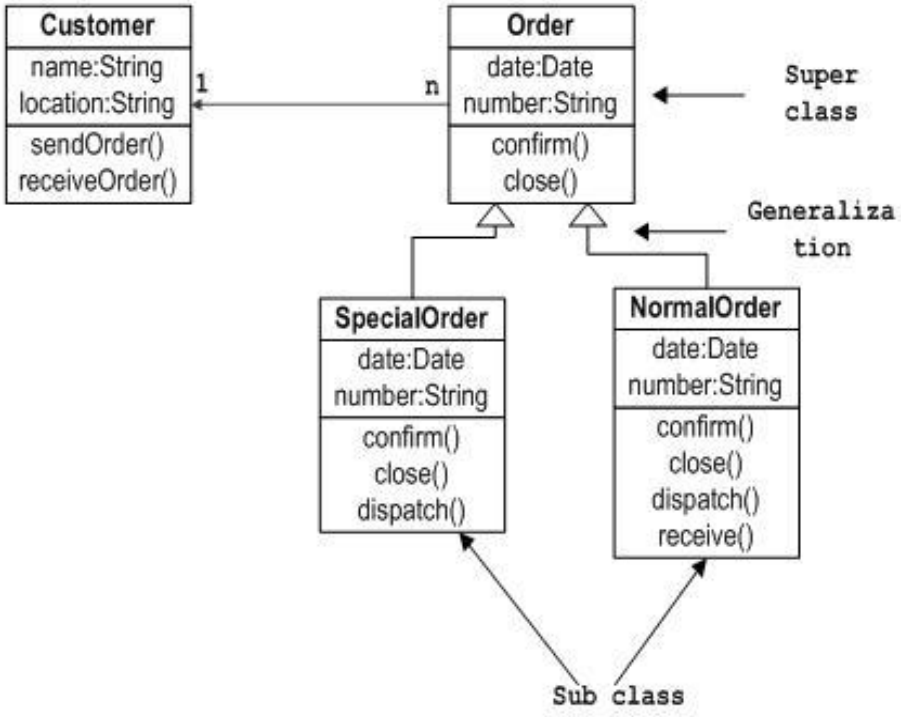
V překladu sekvenční diagram, je to popisem i ukázkou vzájemnou spolupráci objektů, ukazuje posloupnost předávání zpráv. Tento celý soubor předávání zpráv se nazývá scénářem. Je důležitý pro analýzu tak i pro design. Zachycuje dynamický aspekt systému a zobrazuje, jak a kdy spolu objekty komunikují. Zobrazuje také trvání daných instancí, vzhledem k dalším instancím. Na Obrázek 6 je patrná komunikace mezi objektem „Customer“ a objednávkou „Order“, dále ukazuje vytvoření speciální objednávky „SpecialOreder“ objektem „Order“. Svislá přerušovaná čára zobrazuje tzv. čáru života – „Life line“.[3][11][12][14]



Obrázek 6 - Sequence diagram - sekvenční diagram. [14]

2.1.1.3 Class diagram

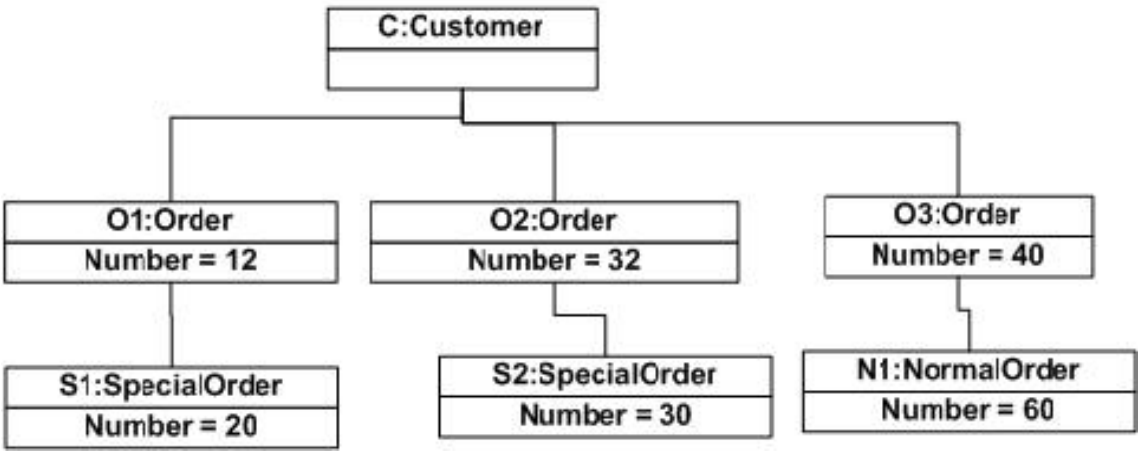
V překladu je možné jej nazývat jako diagram tříd. Popisuje celé třídy v systému a jejich vzájemný vztah. Popisuje systém staticky a navíc je i slovníkem systému, nazývají se podle toho jednotlivé třídy i jejich metody. Diagram tříd není určen pouze pro vizualizaci, je určen především pro programátory jako konstrukt pro tvorbu zdrojového kódu. Na Obrázek 7 je třída „Customer“ a je charakterizována parametry „name“ a „location“ typu „Strin“, dále má metody „sendOrder“ a „reciveOrder“. Je vidět i kardinalitu vztahu k třídě „Order“, která je 1:N. Také vidíme třídy „SpecialOrder“ a „NormalOrder“, které získáme děděním třídy „Order“.[3][11][12][14]



Obrázek 7 - Class diagram - diagram tříd. [14]

2.1.1.4 Object diagram / Instance diagram

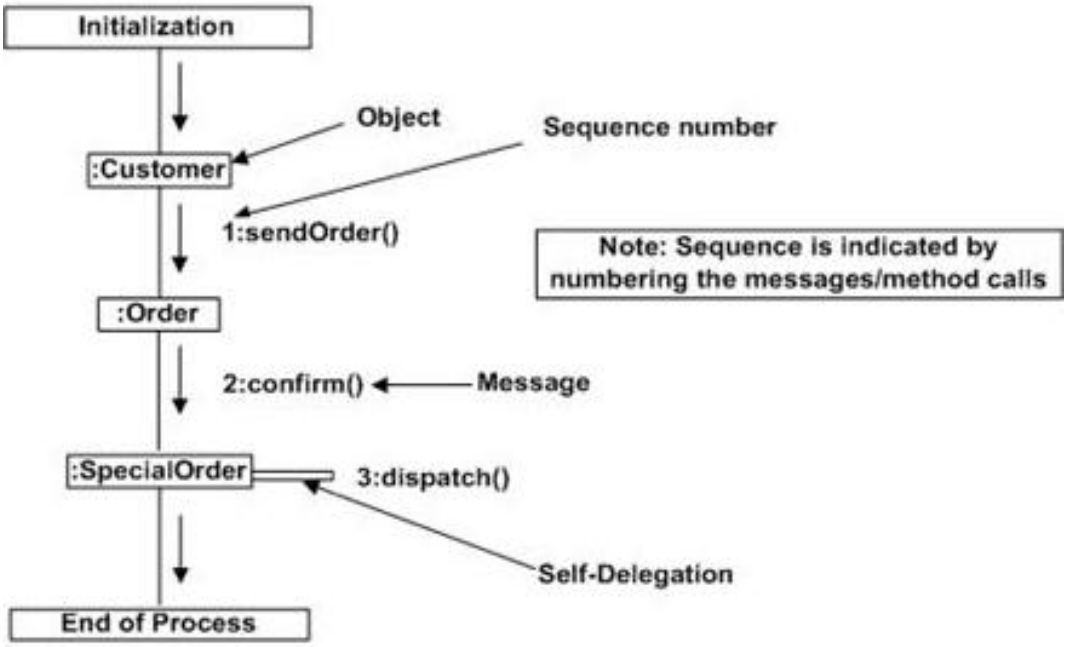
Vyjadřuje vztahy mezi objekty, instancemi. Je odvozený od diagramu tříd. Jedná se statické zobrazení systému, ale již při jeho provozu. Hlavním důvodem tvorby je zobrazení chování systému v daném, konkrétním okamžiku. V tomto modelu se zobrazují určité hodnoty a počty objektů tak, aby reprezentovaly systém a přiblížily se reálnému využití. Viz Obrázek 8, ve kterém je znázorněn uživatel „Customer“ a jeho objednávky „Order“, jež jsou dále blíže specifikované.[3][11][12][14]



Obrázek 8 - Object / Instance diagram – diagram instancí. [14]

2.1.1.5 Collaboration diagram

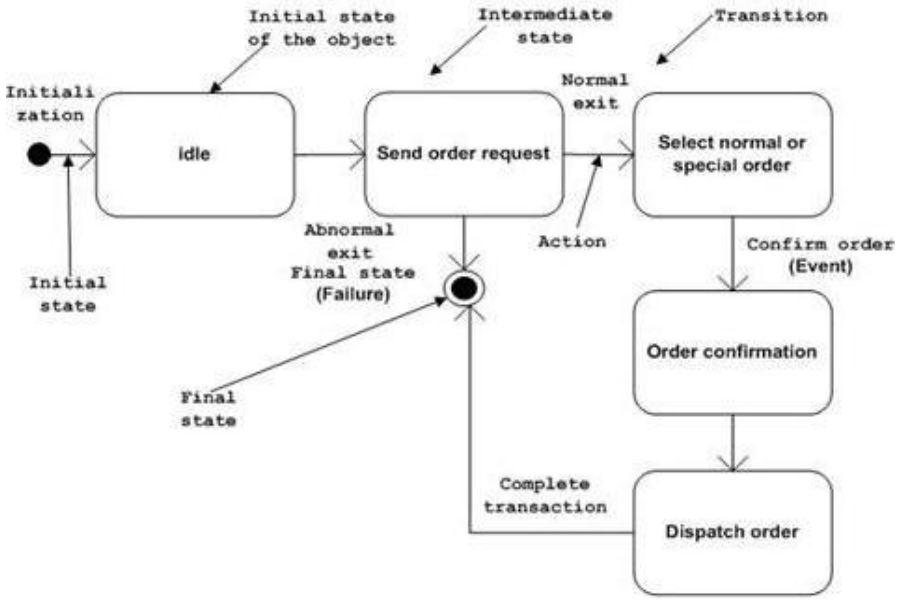
V překladu jej můžeme chápat jako diagram spolupráce objektů. V projektu je zastupitelný sekvenční diagramem. Tvoří se především pro ukázku zasílání zpráv mezi objekty. Vyskytuje se jak v analýze, tak v designu. Používá se pokud je důležitější v projektu organizovanost ve spolupráci objektů. Pokud je pro projekt významnější časová sekvence, používá se sekvenční diagram. Na Obrázek 9 je zobrazeno postupné volání objektů a spouštění jejich metod – od inicializace až po správné ukončení procesu.[3][11][12][14]



Obrázek 9 - Collaboration diagram - diagram spolupráce. [14]

2.1.1.6 Statechart diagram

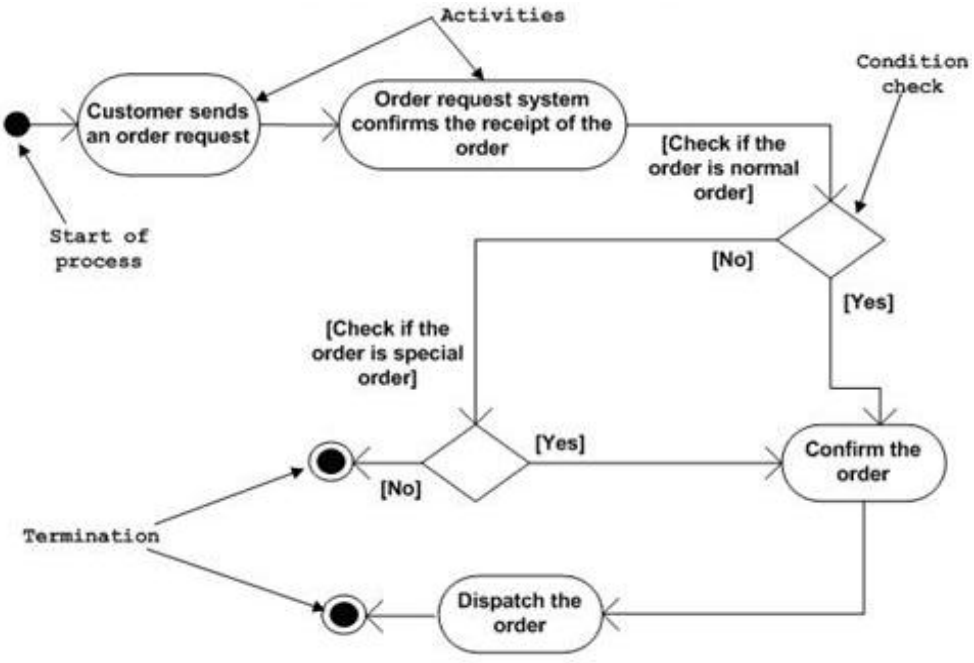
Překládá se jako stavový diagram. Zobrazuje objekty a jejich stavy a změny v přechodech mezi stavy systému. Je důležitý především pro projekty, ve kterých mění stavy uživatel nebo se rozhoduje přímo nějaké zařízení a působí na objekty události. Obecně jej lze chápat jako reaktivní systém reagující na vnitřní nebo vnější události. Na Obrázek 10 je vidět počáteční stav systému a koncový stav, ke kterému se může dojít chybou nebo úspěšným přechodem mezi všemi stavy. Zobrazeny jsou stavy systému, co dělá po provedení dílčích kroků.[3][11][12][14]



Obrázek 10 - Statechart diagram - stavový diagram. [14]

2.1.1.7 Activity diagram

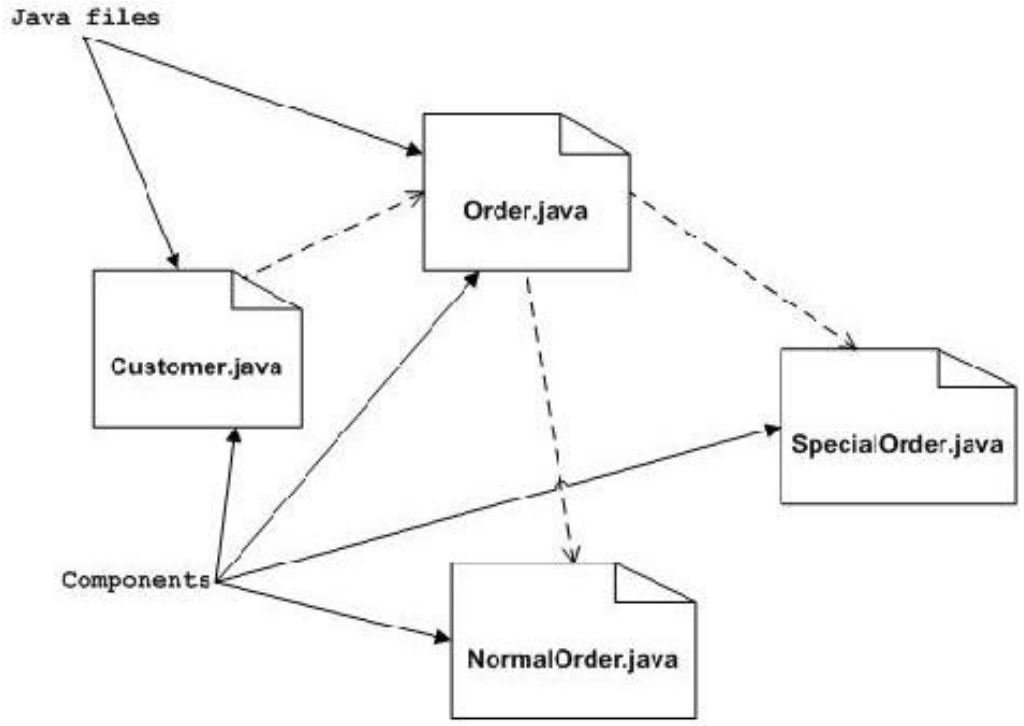
Diagram aktivit systému se vyskytuje jak v analýze, tak v designu. Popisuje sekvenčně přechod aktivit systému. Jejich zobrazení říká, co v danou chvíli má dělat. Diagram se může větvit, může zobrazovat paralelní aktivity, následně se slučovat. Vytváří se od počátku až po standardní ukončení. Na Obrázek 11 můžeme vidět jednotlivé stavy, kterými systém prochází, od spuštění procesu až po jeho ukončení.[3][11][12][14]



Obrázek 11 - Activity diagram - diagram aktivit systému. [14]

2.1.1.8 Component diagram

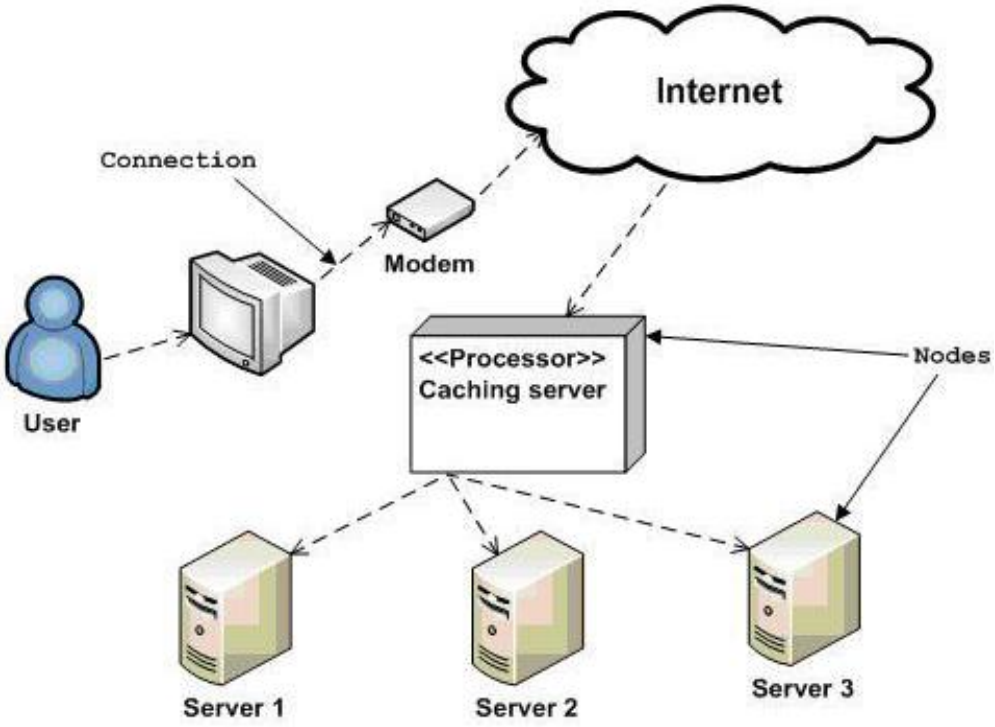
Jedná se o diagram komponent nebo o diagram komponentní. Popisuje komponenty a jejich vazby a vztahy. Nezobrazuje samotnou funkčnost systému, ale zobrazuje komponenty, kterými je systém tvořen. Jde například o knihovny, balíčky, složky atd. Je to staticky implementovaný pohled na systém, který reprezentuje organizaci a vztahy daných komponent. Vyskytuje se v designu. Příklad komponentního digramu můžeme nalézt na Obrázek 12.[3][11][12][14]



Obrázek 12 - Component diagram - diagram komponent. [14]

2.1.1.9 Deployment diagram

Je to diagram zobrazující rozmístění zdrojů. Jedná se čistě o popis HW parametrů a místo uložení dalších zařízení spolu s jejich komunikací. Vyskytuje se především v designu, kde vytváří představu o celkové komunikaci, která bude prováděna. Zobrazuje důležité uzly komunikace a podává přehled o celkové topologii, viz Obrázek 13.[3][11][12][14]



Obrázek 13 - Deployment diagram - diagram rozložení zdrojů. [14]

2.2 Metodiky vývoje softwaru

První metodiky započaly s rozvojem modulárního programování, jednalo se o několik rad, doporučení. Postupně s rozvojem programovacích technik a rozvojem nástrojů pro programování začaly rozšiřovat a využívat další nástroje. Čímž metodiky narůstaly na objemu. Metodiky jsou postupy, doporučení a konstrukce několika generací programátorů, které jsou unifikovány. V současnosti existuje mnoho metodik, které jsou vhodné pro různé typy projektů. Pro výběr nejvhodnější metodiky je možné využít vícekriteriální systém hodnocení a výběru metodik Methodology Evaluation System (METES).[8] Nejrozšířenější metodikou a ve většině společností standardem je UP, který určuje co, kdo a kdy má při vývoji SW za úkol. Charakterizovány jsou dva hlavní směry, a to rigorózní metodiky a metodiky agilní. Rigorózní metodiky vycházejí z možnosti procesy při tvorbě SW řídit, plánovat a měřit. Mají snahu o přesné definování procesů, činností a produkty. Aby popsaly vše nutné, bývají celkem objemné. Agilní metodiky si zakládají na větší míře kooperace s uživateli, kdy je možné měnit požadavky průběžně. Na začátku jsou definovány pouze hrubé požadavky. Přístupy jsou podobné rychlému vývoji aplikací (RAD Rapid Application Development), ale liší se především v kvalitě zpracování návrhu, která je nezbytná pro realizaci změn. Jedná se iterativní činnosti, kdy návrh je prováděn i

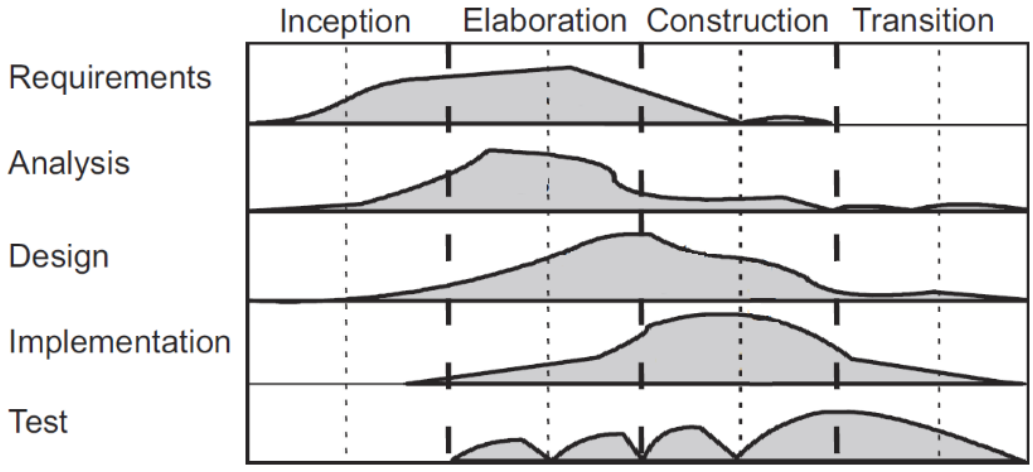
v průběhu projektu. Dalším zásadním požadavkem je Jednoduchost řešení a samo organizovanost pracovních týmů.[8][11][13]

2.2.1 OPEN metodika

Je to rigorózní metodika, definující rámec OPEN Process Framework (OPF), z kterého je možné generovat specifické instance pro danou společnost – konstrukce procesu. Obsahuje skupinu metatříd seskupených do pěti skupin, a to na pracovní jednotky, pracovní produkty, producenty, etapy a jazyk. To vytváří knihovnu komponent Open. Třetí částí jsou činnosti, které určují co je zapotřebí dělat, ale neurčují způsob provedení. Mají malý rozsah přizpůsobený tak, aby jej mohl provádět jednotlivec nebo malý tým v co nejkratším čase. Využívá modelace UML, nebo vlastní OPEN Modeling Language (OML).[8][13]

2.2.2 Metodika unifikovaného procesu

Metodika podložena rigorózním přístupem. Je založena na praktikách iterativního vývoje, řízení požadavků, využívání komponentové architektury, vizuálního modelováni, kontroly kvality SW a řízení změn. Je to standardem ve vývoji SW. Strukturu vývoje softwaru tvoří čtyři cykly. První je zahájení (Inception), dalším je rozpracování (Elaboration), následuje konstrukce (Constructin) a poslední fází je zavedení (Transition). V každé fázi se opakují s různou intenzitou činnosti, jakými jsou požadavky (Requirements), analýza (Analysis), návrh (Design), implementace (Implementation) a testování (Test). Na Obrázek 14 je graficky znázorněno množství práce, které se provádí. Zároveň zobrazuje fáze, ve kterých je činnost prováděna a v jakém objemu.[3][8][11][13]



Obrázek 14 - Rozložení práce v jednotlivých fázích projektu. [8]

Každá fáze je zakončená milníkem (milestone). Je to soubor požadavků, které je nutné splnit, nežli bude započata další iterace. V počáteční fázi je definováno, jaké jsou cíle projektu, požadavky, sestavení harmonogramu pro projekt, vytvoření modelu, nebo i vytvoření jednoduchého prototypu. Fáze se ukončuje rozhodnutím, zda je projekt možné realizovat v daných podmínkách a s dostupnými zdroji. Ve fázi rozpracování je cílem definovat architekturu systému, vytvoření prototypu pro ověření architektonických principů, umožnění upřesnění plánu realizace. Definují se znovu použitelné komponenty. V konstrukční fázi je proveden návrh a realizace systému, včetně testování. Ve fázi zavedení je zajištění použitelnosti SW, školení uživatelů, předání dokumentace a vytvoření podpory. UP má i specifika, která se týkají přímo členů pracujících na projektu. Nazývají se role. Obsahuje role jako vedoucí projektu, uživatel, specialista na požadavky, správce databáze, hlavní programátor, analytik, vývojář, dokumentátor, tester, analytik testů, manažer testů, návrhář testů. U činností je to různé, liší se podle fáze, s jakým nasazením se na nich bude pracovat. [3][8][11][13]

Cílem je specifikace požadavků vedených uživateli. Provádí se rozhovory a tvoří se dokumenty aplikační oblasti. Výstupem jsou případy užití, aplikační slovník a kompetenční otázky. Modelují se případy užití. Využívá se standardu UML. V činnosti zabývající se analýzou je úkolem detailní rozpracování a strukturování požadavků. Výstupem je doplněný diagram užití, diagramy tříd, diagramy aktivit a referenční výkladový slovník. Činnost specializující se na návrh, jejím cílem je vytvoření struktury pro položky výkladového slovníku. Vstupem je referenční výkladový slovník, diagramy tříd a aktivit. Tedy modelování pojmů, hierarchických a specifických vztahů. V části implementace je cílem zakódování do formálního jazyka. V činnosti pro testování se provádí kontrola konzistence, verifikace pokrytí požadavků a pragmatická kvalita SW.[1][3][8][11][13]

2.2.3 Metodika Enterprise Unified Process

Rozšiřuje metodiku UP ve dvou směrech. Rozšiřuje se na úroveň celé organizace, jež zahrnuje procesy realizované skrz projekty. Druhý směr rozšíření jsou činnosti zabývající se provozem, údržbou systému a jeho odstranění z provozu.[1][3][8][13]

2.2.4 Metodika Dynamic System Development Method

DSDM patří mezi agilní metodiky. Její zaměření reprezentuje devět pilířů, které charakterizují a korespondují s myšlenkou agilních metodik. Jedná se o aktivní zapojení uživatele, vytvořeny jsou týmy s rozhodovací pravomocí, časté dodávky produktů, iterativní a inkrementální postup, změny během vývoje, definice požadavků na povrchní úrovni, spolupráce mezi členy týmu a cílem pro přijetí dodávky je podpora podnikových cílů (růst společnosti). Rozděluje a modeluje v rámci třech hlavních fází. První je funkční model (Functional Model), návrh (Design and Build) a implementace (Implementation). Předchází jim ještě studie proveditelnosti (Feasibility Study) a byznys studie (Business Study). Funkční model zahrnuje sběr a prototypování funkčních požadavků. Většina požadavků je dokumentována jako prototypy. Ve fázi návrhu jsou prototypy zpodrobňovány a je navrženo řešení. V implementační fázi je realizováno navržené řešení, zhodnocení projektu a školení uživatelů.[1][8]

2.2.5 Metodika Adaptative Software Development

V agilní metodice adaptativního softwarového vývoje (ASD) je hlavní přístup ke změnám, které nastávají. Metodika ASD má iterační životní cyklus se třemi fázi - spekulace (Speculate), spolupráce (Collaborate) a učení (Learn). Ve fázi spekulace se provádí dotazy na změny, experimentuje a zkoumá odchylky od plánu a bere je jako příležitosti k učení. Ve fázi spolupráce spolu kooperují celé týmy na řešení daných odchylek. V poslední fázi učení se prověřují předešlé zkušenosti, jak úspěchy, tak i neúspěchy. Kontroluje se kvalita řešení z pohledu zákazníka, technologie, fungování týmů, použitých praktik a celkovým stavem projektu.[1][8]

2.2.6 Metodika Lean developement

Patří mezi agilní metodiky, je založena na konceptu dynamické stability. Řídí se několika pravidly. Základním kamenem metodiky je práce s třetinovými náklady, to znamená omezení pro lidskou pracovní sílu, čas, investice do nástrojů a metod a námahou pro přizpůsobení se tržnímu prostředí. Mezi desatero pravidel patří odstranění zbytečného – zbavení se všeho, co nepřidává konečnému produktu hodnotu jako jsou například dokumenty, diagramy. Minimalizování zásob na úroveň co největší abstrakce, pokud to není přímo součástí systému. Maximalizování toku iterativním vývojem. Vývoj je „tažen“ poptávkou, což znamená co nejpozdější rozhodování a přizpůsobení se tak požadavkům

zákazníka. Vyhnutí se neúspěchu projektu z hlediska neúplných nebo nesprávných požadavků řeší odsouhlasením konečného uživatele, i když je riziko neúplných požadavků vysoké. Proto je zavedena zpětná vazba, která doplňuje požadavky v průběhu vývoje. Z firemního hlediska je důležité partnerství s dodavateli a vytvoření podmínek pro motivaci zlepšovat procesy při vývoji SW.[8][13]

2.2.7 Metodika Featrue Driven Developement

Metodika FDD patřící mezi agilní a iterativní pracovní postupy, je řízena užitnými vlastnostmi produktu. Nejdříve se vytvoří celkový model, následně se pokračuje posloupností dvoutýdenních iterací, na nichž se provádí návrh i realizace pro jednotlivé užitné vlastnosti (Feature). Užitná vlastnost je chápána jako výsledek užitečný z pohledu zákazníka, ten je měřitelný, srozumitelný a realizovatelný. Je velice široce popsána a staví výsledný produkt nad použité procesy vývoje. Postup metodik začíná vytvořením celkového objektového modelu (Develop an Overall Model), následuje sestavení seznamu užitných vlastností (Build a Features List), následně se provede plán pro užitnou vlastnost (Plan by Feature), návrh (Design by Feature) a její realizace (Build by Feature).[8][13]

2.2.8 Metodika Scrum

Metodika Scrum nahlíží na vývoj SW jako na empirický proces. Je zaměřen na řízení projektů. Vývoj probíhá ve třiceti denních iteracích – sprintech (Sprint), které jsou zakončeny dodáním vybraných případů užití. Pracovním postupem je iterace čtyřech fází. První fáze (Planing Phase) má za úkol specifikace prvních požadavků, plánování dodávek a uvedení architektonických a byznys vizí. Druhá fáze je vynášecí (Staging Phase), ve které jsou připojeny nefunkční požadavky. Ve fázi vývoje (Development Phase) dodává tým funkcionalitu a na konci fáze ji předvede. Ve fázi dodávky (Release Phase) je výsledný produkt předán uživatelům. Nemá pevné stanoviska, proto využívá každodenní, patnácti minutové porady (Scrum Meetings), sloužící pro koordinaci a integraci práce všech zaměstnanců. Na těchto poradách každý sdělí, které položky dokončil, jaké jsou jeho nové úkoly a jaké vidí překážky a omezení ve splnění úkolu.[1][8][13]

2.2.9 Metodika Extrémního Programování

Jedná se o agilní metodiku vycházející ze standardních postupů při vývoji SW. Tyto postupy dovádí do extrému. Osvědčené metody se provádí neustále. Pokud se osvědčí revize zdrojového kódu, bude se revidovat ustavičně. Testování se provádí pořád, testují se

jak jednotky, tak i funkcionality. Při důležitosti architektury se provádí nepřetržitá úprava a rozpracování částí. Při osvědčení iterace se interval zkrátí a provádí se po minutách, hodinách. Důležitým aspektem extrémního programování (XP) je pohled na náklady spojené se změnami.[2][8][13]

2.3 Návrhové vzory

První pojem „návrhový vzor“ (Design Pattern) byl použit v roce 1987, kdy skupina programátorů, ve snaze pomoct začínajícím kolegům a jejich vedení, představila dokument „Using Pattern Languages for Object-Oreinted Programs“. Jednalo se sice o jazyk Smaltalk, ale dalším vývojem se z návrhových vzorů stal nástroj nezávislý na programovacím jazyku. Návrhové vzory nejsou hotovým řešením připraveným k okamžitému použití, je to model, který je potřeba poznat a přizpůsobit konkrétní situaci. Jsou to možnosti, které dovolují vytvořit kvalitní objektově orientovaný produkt. Návrhové vzory slouží jako inspirace pro myšlení. V roce 1994 vydala skupina programátorů, známých pod zkratkou GoF, knihu, sbírku „Design Patterns: Elements of Reusable Object-Oriented Software“, která je aktuální i v současné době. Návrhový vzor obsahuje čtyři části, kterými je charakterizován. První částí je jeho název, který definuje vzor. Pod jeho názvem jej můžeme prezentovat jako další stupeň abstrakce, bez bližších detailů. Druhou částí je popis problému, ve které je napsané kdy se má daný návrhový vzor použít, vysvětlení kontextu, některé vzory mají uvedeny podmínky, které musí byt splněny pro použitelnost daného vzoru. Ve třetí části je popsáno jaké prvky se používají, jaká je konstrukce. Charakterizuje vztahy, povinnosti a spolupráci objektů. Obsahuje abstraktní popis tříd a objektů pro řešení situace. V poslední části je uvedeno, jaké jsou důsledky a kompromisy uvedeného vzoru, informace o časové a implementační složitosti s vhodným využitím prostředků. Původně jsou rozděleny vzory do třech druhů – Tvořivé vzory (Creational), Strukturální vzory (Structural) a Vzory chování (Behavioral). Vzory tvořivé definují a ukazují, jakým způsobem lze a je vhodné vytvořit požadované instance dle jejich druhu, počtu, nebo funkce. Vzory strukturálního typu jsou používány pro vytváření spolupráce mezi většími částmi systému, definují převážně způsoby komunikace mezi objekty. Vzory chování, poukazují na funkčnost vytvořených instancí, co mají za úkol provést a jakým způsobem.[1][4][10][11][15]

3 ŠKOLNÍ PROJEKTOVÁ VÝUKA

Tvorba SW produktu je činnost praktická, většinou prováděna týmy, složených z různých členů, jejichž výsledkem je funkční program pro zadavatele. V prostředí školy je kladen důraz především na učení, osvojení si znalosti. V projektové výuce jde především o řešení komplexnějších úloh. Tyto výukové záměry a plány mají také širší praktický dosah. Charakteristická je zejména tím, že překračuje hranice školy. Zmíněné hranice mohou být posunuty jak ve směru přírodovědném, tak do společenské komunity nebo do výrobního procesu. Přijímáním této formy výuky na sebe berou určitou zodpovědnost se začleněním do životní praxe. Protože tvorba SW zahrnuje mnoho předmětů, které je nutné zvládnout pro dosažení požadovaného výsledku, tak i projektová výuka zasahuje a sdružuje předměty přirozenou cestou, protože cílem je vyřešení dané situace, jež je přebrána ze životní reality. Projektovou výuku lze datovat z 18. století, první pokusy pocházejí z dílny J. Deweye a W. H. Kilpatrica o dvě stě let později. Vymezení pojmu projekt můžeme chápat jako: *„Kompletní praktickou úlohu spojenou se životní realitou, kterou je nutno řešit teoretickou i praktickou činností, která vede k vytvoření adekvátního produktu.“[16]* Dále je možné dělit projekt na několik fází. Prvním je stanovení cíle, to má zajistit vhodnost a realizovatelnost záměru vzhledem k daným podmínkám. Významnou roli hraje motivace žáků. Žáci se musí s daným projektem a úkolem ztotožnit a přijmout je za své. Následuje další fáze, a to vytvoření plánu řešení. Představuje rozhodující okamžik, který určí výsledek. Proto záleží na společné diskuzi postupu a výběrů úkolů pro žáka nebo skupinu. V této fázi se provádí kalkulace nad použitým materiálem a zajištění nezbytných zdrojů. Stanovují se dílčí odpovědnosti za úkoly a jejich prezentace ve finální podobě. Pokud jsou možnosti, je velmi vhodné zpřístupnit celkový plán všem po celou dobu trvání projektu, pro jeho průběžnou kontrolu. Následující fáze „Realizace plánu“ je naplněna kontrolou průběhu, dle vypracovaného plánu. Probíhá realizace všech aktivit spojených s vypracováním všech částí projektu. Je vyžadována odpovědnost ze strany žáků nad plněním, vnímáním, pozorováním a experimentováním. Závěrečnou fází je vyhodnocení. Opírá se zejména o sebekritiku a objektivní posouzení přínosu všech jedinců, kteří se podíleli na řešení. Důležitou součástí je prezentace výsledků veřejnosti, rodičům tak, aby byl u žáků vytvořen pocit uspokojení, a tím i dosažení sebedůvěry ve své schopnosti. Jak v případě projektů provozovaných v praxi, tak ve školních projektech, se stanovuje čas, který odpovídá zvoleným cílům a vybranému tématu. Rozsah může být krátkodobý (dvě až několik hodin), střednědobý (realizace během dvou dnů), dlouhodobý (několik týdnů) a

mimořádně dlouhodobý (doba od několika týdnů i měsíců, realizovaný s běžnou výukou). Realizace projektu může být provedena několika způsoby. Práce může být prováděna skupinově, individuálně nebo nejčastěji kombinovaně.[16][17]

4 VÝZKUM DOTAZNÍKEM

Dotazníkový výzkum se provádí především z potvrzení hypotéz a získání informací. Obecně je dotazník vymezován jako písemně předem připravený soubor otázek, na které se získávají písemné odpovědi. Dotazník může zkoumat vnější jevy nebo vnitřní postoje dotazovaných subjektů. Dotazník je dáván vždy konkrétním osobám, a tyto si vybírá dotazovatel. Při sestavování dotazníku je vhodné dbát na sestavení otázek tak, aby zkoumaly co nejvíce objektivní postoje, nežli subjektivní, to se týká i vyhodnocování dotazníku. V dotazníku se položky, chápané nejen jako otázky ale i příkazy, objevují ve dvou formách, a to uzavřené položky (strukturované) a otevřené (nestrukturované). Na nestrukturované otázky respondent odpovídá vlastními slovy nebo podle návodu. Při strukturovaných položkách je vždy zobrazena možná odpověď nebo několik odpovědí. Dotazník by měl splňovat tři základní kritéria, jimž jsou validita, reliabilita a praktičnost.[18]

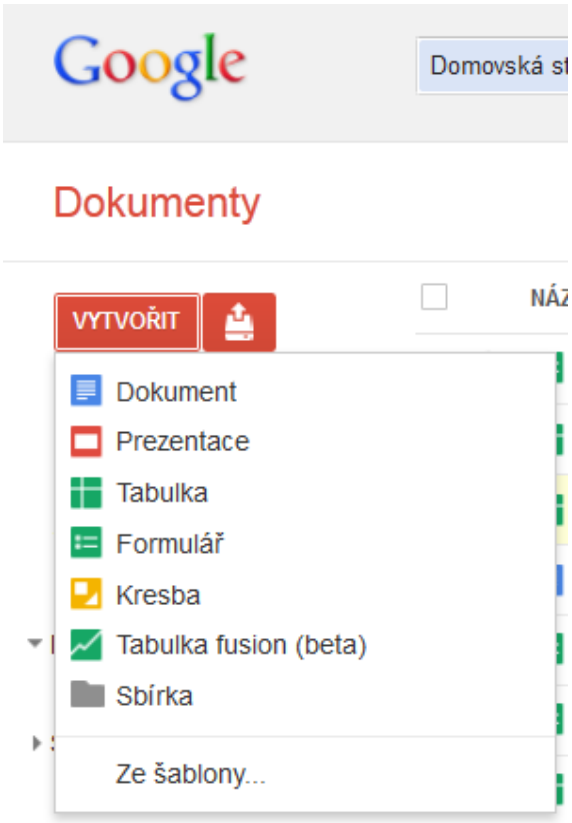
II. PRAKTICKÁ ČÁST

5 DOTAZNÍK

Pro analýzu základních a středních škol jsem si zvolil formu dotazníku. Dotazníkem, je možné v krátké době dotázat velké množství osob. Použitý dotazník má za cíl zjistit, jakým způsobem a v jakém množství je projektová výuka začleněna na ZŠ a SŠ. Protože projektová výuka je nejblíže praktickému využití a dává možnost vyzkoušet žákům budoucí pracovní prostředí. Praktickou výukou se eliminuje přechod ze školního prostředí do prostředí pracovního. Tímto žák získá nejen potřebné teoretické znalosti, ale i zkušenosti, které můžou umožnit nalezení lepšího pracovního uplatnění. Další zkoumanou částí bylo vytváření SW projektů na školách. Jakým způsobem pracují, zda tvoří skupiny, nebo se žáci účastní mimoškolních soutěží zaměřených na programování. Následně kolik SW produktů žáci vytvoří během školního roku a kolik času věnují jednotlivým fázím tvorby.[16][18]

5.1 Tvorba dotazníku

Dotazník byl vytvořen za použití online kancelářské sady společnosti Google, pojmenovanou „Google Docs“. V části formuláře je možné vytvořit online dotazník.



Obrázek 15 – „Google Docs“ – Formulář. [19]

Dotazník byl vytvořen pro zjištění, zda probíhá projektová výuka na středních (SŠ) a základních školách (ZŠ). Dalším bodem zkoumání bylo, zda žáci těchto škol tvoří SW projekty, a jakým způsobem se věnují jednotlivým částem tvorby. Celkový dotazník je uveden jako „PŘÍLOHA P I: DOTAZNÍK“. Data získaná z dotazníku lze upravovat online (Obrázek 16) nebo exportovat do různých souborů (Obrázek 17). Otázky po publikaci již nebyly modifikovány. Celková úprava a tvorba dotazníku probíhala za použití služby společnosti Google (Obrázek 18) v prohlížeči „Mozilla Firefox“.[19][20]

Tvorba softwarového projektu na ZŠ a SŠ.

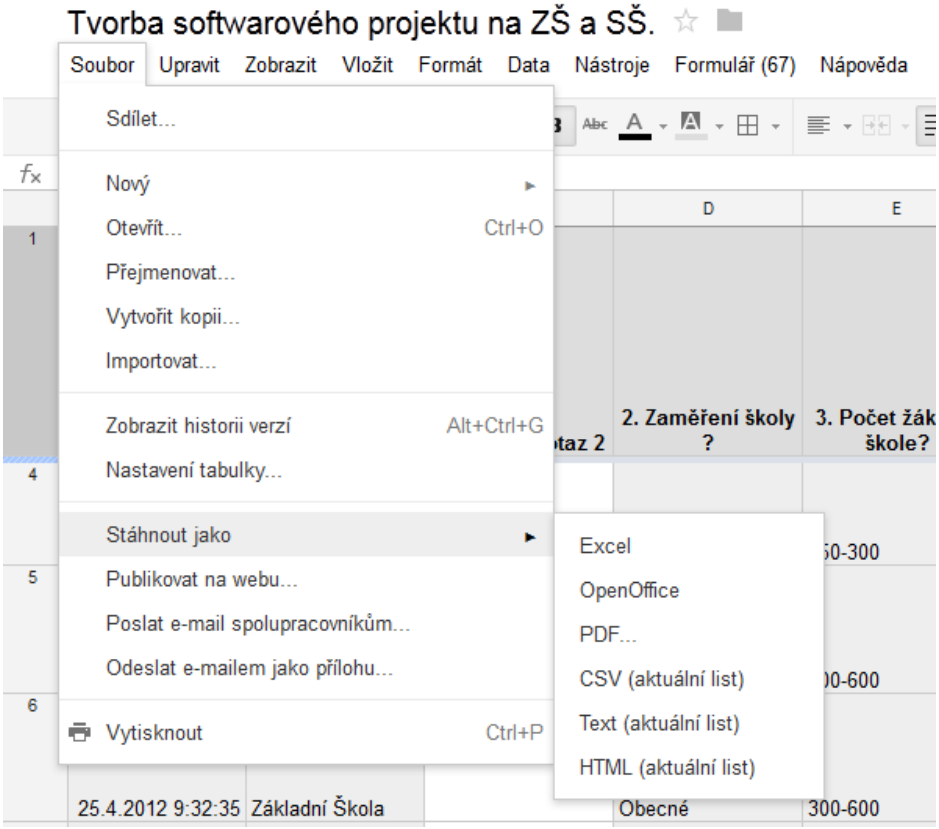
Soubor Upravit Zobrazit Vložit Formát Data Nástroje Formulář (67) Nápověda Poslední úprava provedena před 4 dny uživatelem m.zmeskal

Časová značka

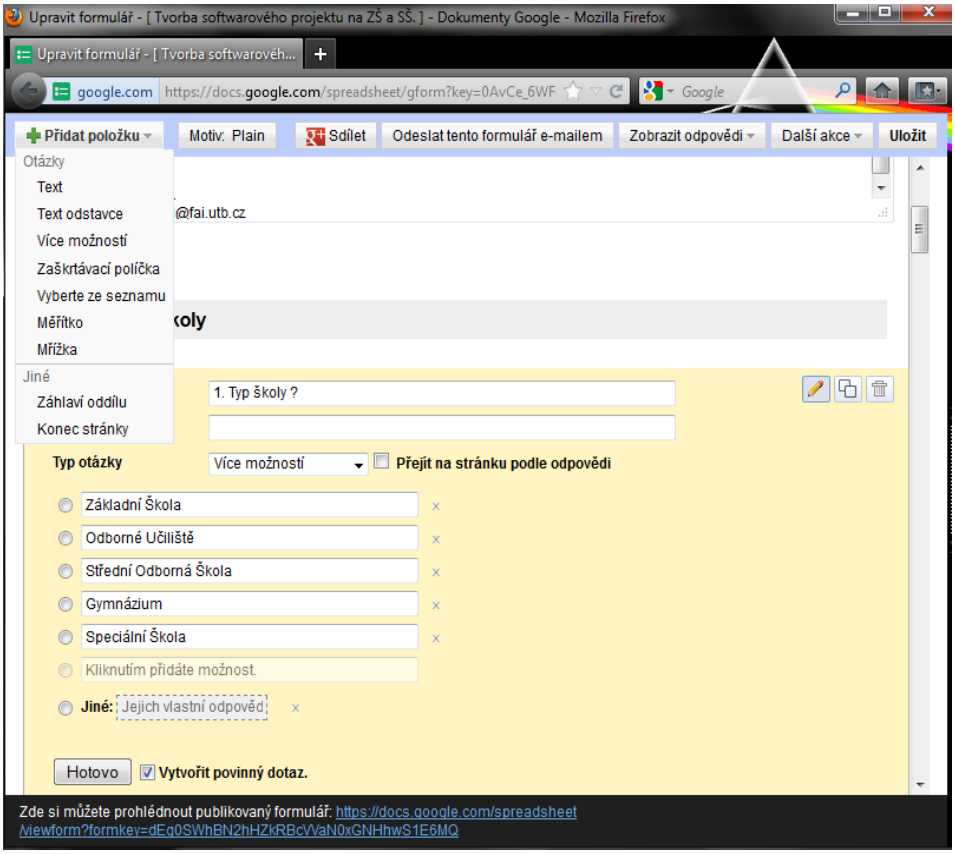
	A	B	C	D	E	F	G	H
1	Časová značka	1. Typ školy ?	Ukázkový dotaz 2	2. Zaměření školy ?	3. Počet žáků na škole?	4. Doplňte dle vlastních zkušeností : [Obecné vybavení školy - ve všech učebnách]	4. Doplňte dle vlastních zkušeností : [Vybavení odborných učeben]	4. Doplňte dle vlastních zkušeností : [Vybavení učeben pro Informatiku]
4	25.4.2012 8:05:09	Základní Škola		Obecné	150-300	Dobré	Dobré	Dobré
5	25.4.2012 8:31:53	Základní Škola		Obecné	300-600	Dostatečné	Dostatečné	Dobré
6	25.4.2012 9:32:35	Základní Škola		Obecné	300-600	Dostatečné	Dobré	Dobré
7								

List1

Obrázek 16 - Online dotazníková data. [19]



Obrázek 17 - Export do různých formátů. [19]



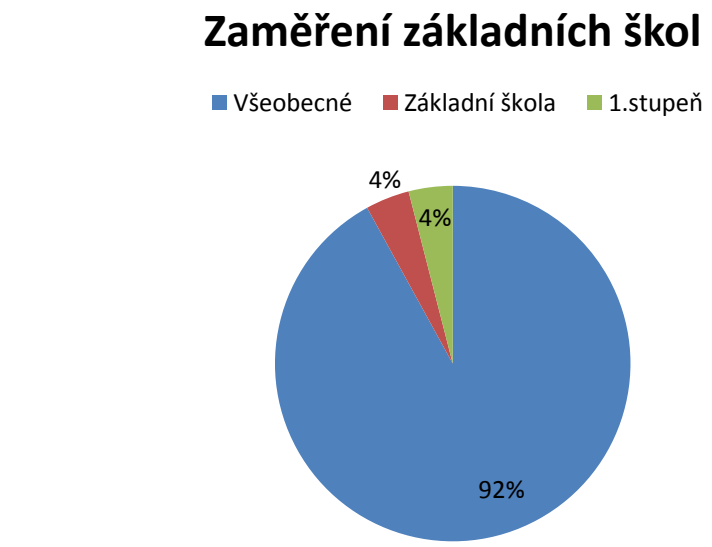
Obrázek 18 - Úprava formuláře v prostředí Mozilla Firefox. [19][20]

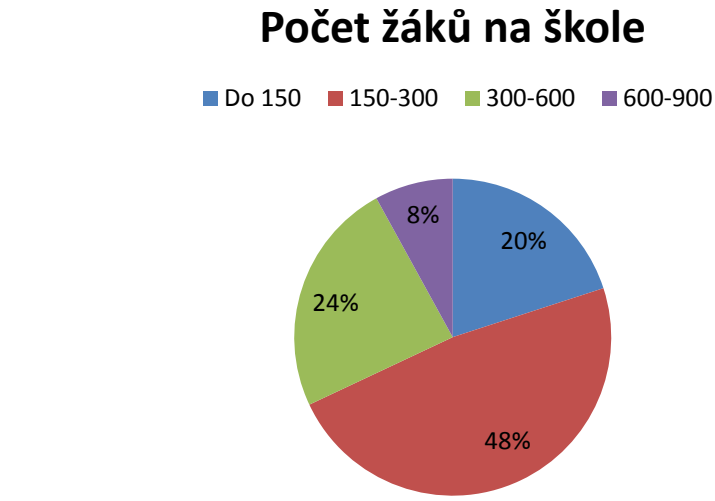
5.2 Vyhodnocení dotazníku

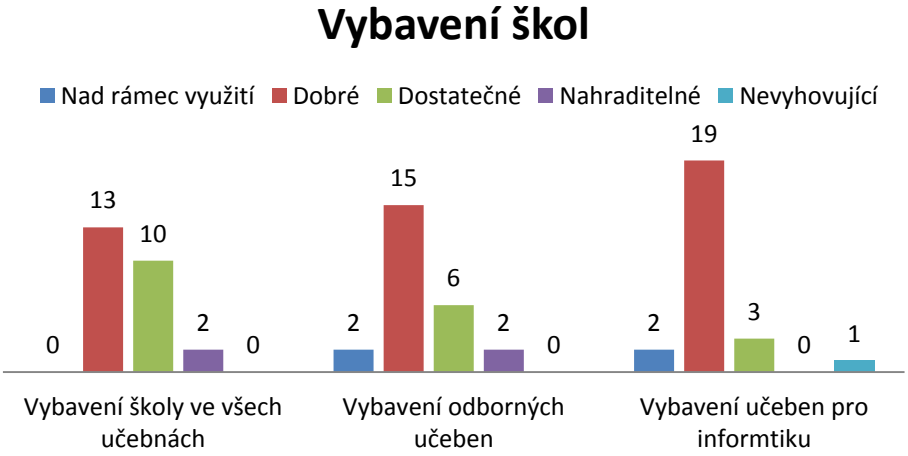
Dle dotazníku rozděleno i vyhodnocení na základní školy a střední školy včetně gymnázií. Vyhodnocení je uvedeno po jednotlivých otázkách v grafickém zobrazení.

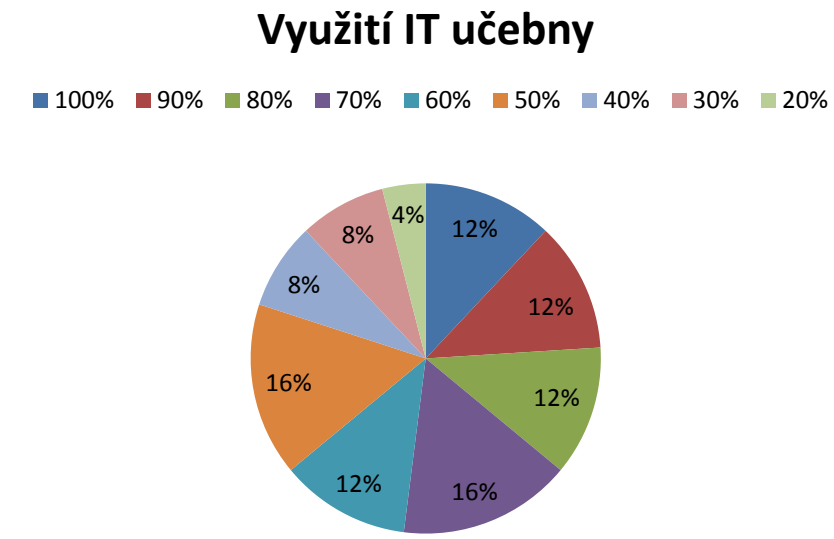
5.2.1 Vyhodnocení – základní škola

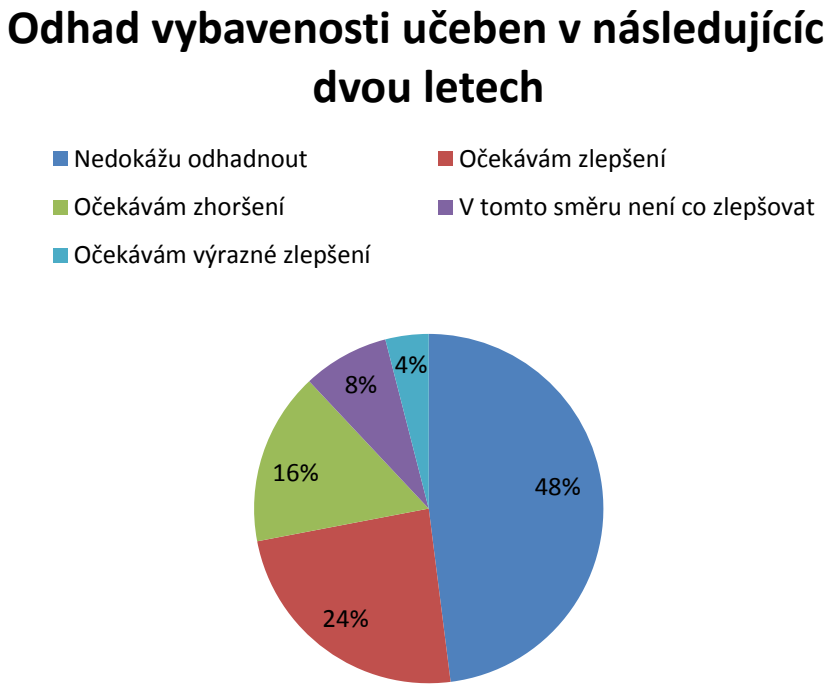
Číselně vyjádřeno, ze základních škol se zúčastnilo 8,7% oslovených institucí. Zaměření dotázaných škol je všeobecné, pouze jedna škola, která se zúčastnila, provozuje výuku pouze na prvním stupni. Školy mají z větší části mezi sto padesáti až šesti sty žáky. Vybavení učeben hodnotí pedagogové jako dobré a dostatečné. Vybavení odborných učeben hodnotí jako dobré, stejně tak hodnotí i vybavení učeben pro informatiku. Učebny pro informatiku jsou průměrně využity, po stránce využití vybavení, ze šedesáti pěti procent. V příštích dvou letech očekává více než čtvrtina (28%) zlepšení současného stavu vybavení a téměř polovina (48%) nedokáže odhadnout. Projektovou výuku na základních školách využívá šedesát osm procent učitelů a věnují ji průměrně 29,4% z běžné výuky. Výhodou je, že 65% dotázaných pedagogů má vyčleněno mimo běžnou výuku prostor právě pro výuku projektovou. Žáci základních škol se neúčastní žádných informatických olympiád. Krátkodobým plánem realizuje projektovou výuku 62 % a dalších 25 % dotázaných provádí projektovou výuku se střednědobým plánem. Žáci při projektové výuce jsou podle vyučujících více motivováni, jsou aktivní, tvoriví a iniciativní. Při práci ve skupinách pracují ve dvou až třech žácích. Obvykle za rok vytvoří tři až šest softwarových produktů. Co se týká SW produktů, žáci základních škol, dle odpovědí učitelů, z 92% žádné nevytváří. Veškeré odpovědi jsou níže zpracovány i s grafickou modifikací.

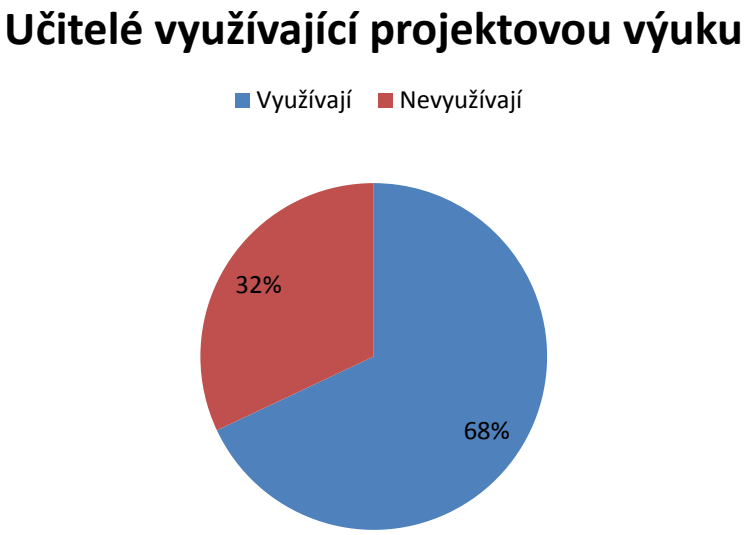

Zaměření základních škol
Všeobecné
Základní škola
1.stupeň
4%
4%
92%

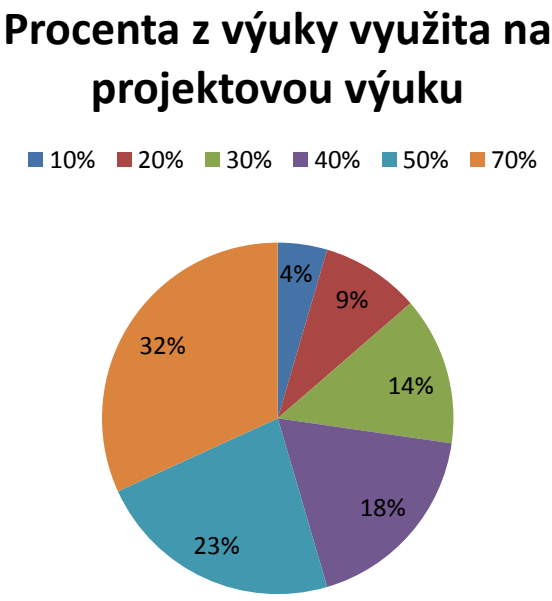

Počet žáků na škole
Do 150
150-300
300-600
600-900
8%
20%
24%
48%

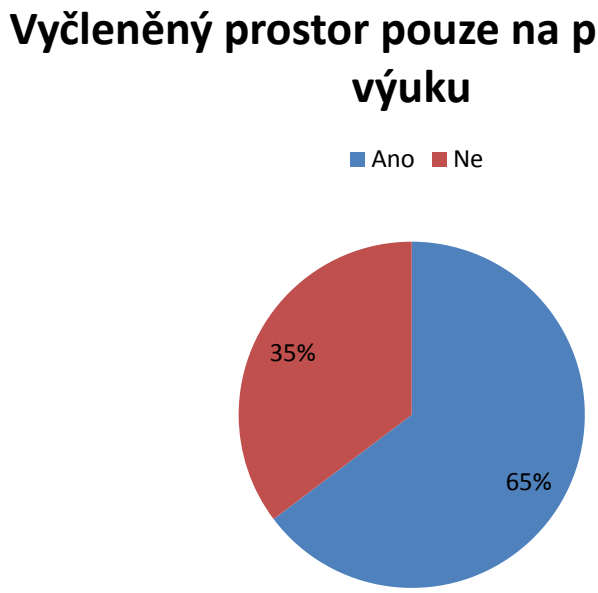

Vybavení škol
Nad rámec využití
Dobré
Dostatečné
Nahraditelné
Nevyhovující
0
13
10
2
0
2
15
6
2
0
2
19
3
0
1
Vybavení školy ve všech učebnách
Vybavení odborných učeben
Vybavení učeben pro informtiku

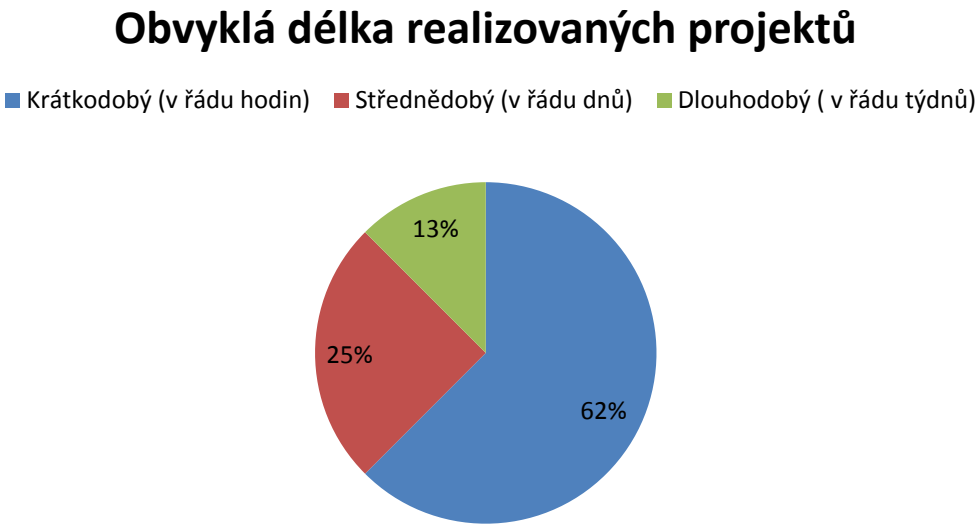

Využití IT učebny
100%
90%
80%
70%
60%
50%
40%
30%
20%
12%
12%
12%
16%
12%
16%
8%
8%
4%

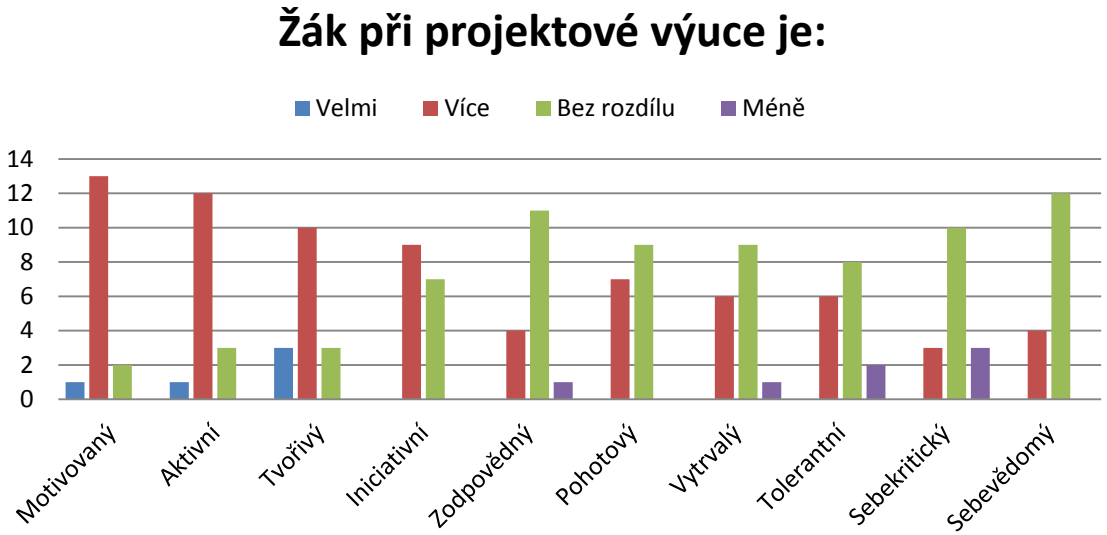

Odhad vybavenosti učeben v následujících dvou letech
Nedokážu odhadnout
Očekávám zlepšení
Očekávám zhoršení
V tomto směru není co zlepšovat
Očekávám výrazné zlepšení
48%
24%
16%
8%
4%

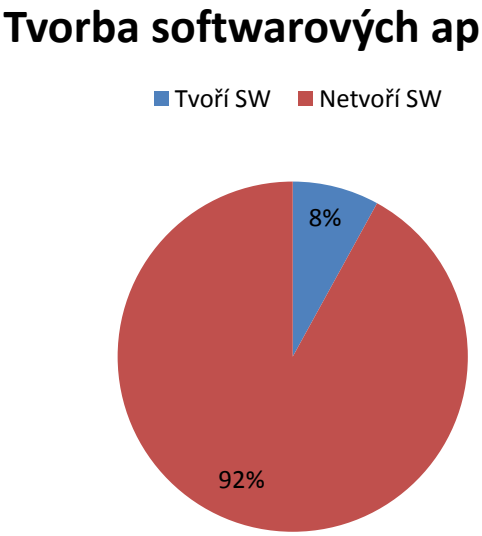

Učitelé využívající projektovou výuku
Využívají
Nevyužívají
32%
68%


Procenta z výuky využita na projektovou výuku
10%
20%
30%
40%
50%
70%
4%
9%
14%
18%
23%
32%


Vyčleněný prostor pouze na projektovou výuku
Ano
Ne
35%
65%


Obvyklá délka realizovaných projektů
Krátkodobý (v řádu hodin)
Střednědobý (v řádu dnů)
Dlouhodobý (v řádu týdnů)
13%
25%
62%

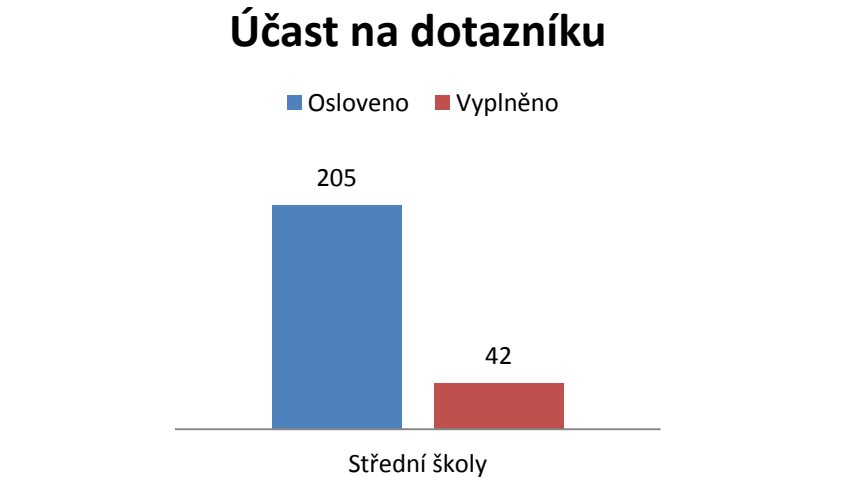


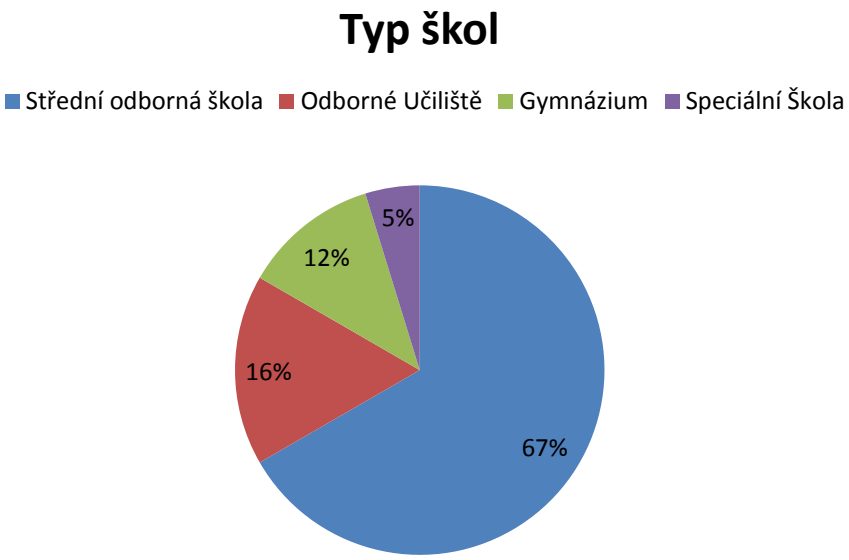


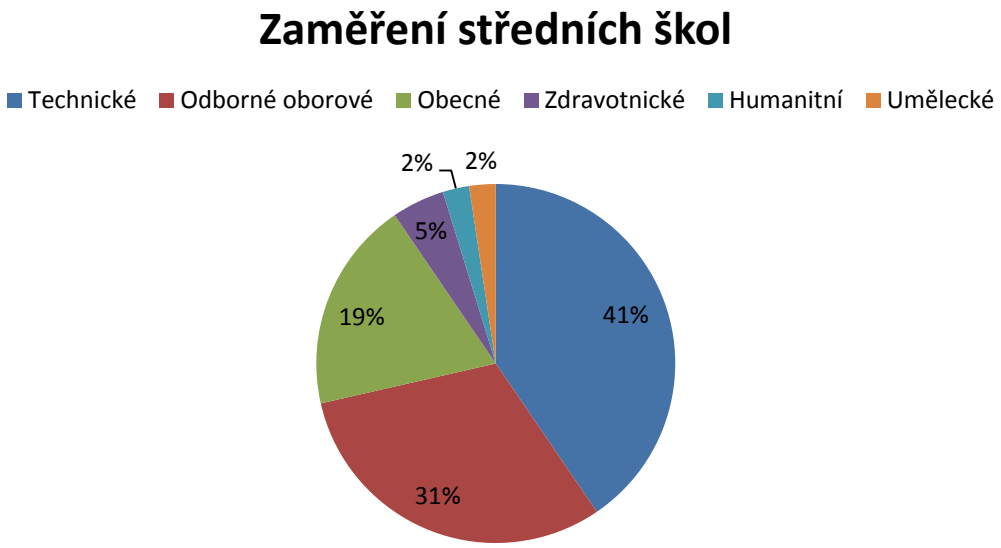
5.2.2 Vyhodnocení – střední škola

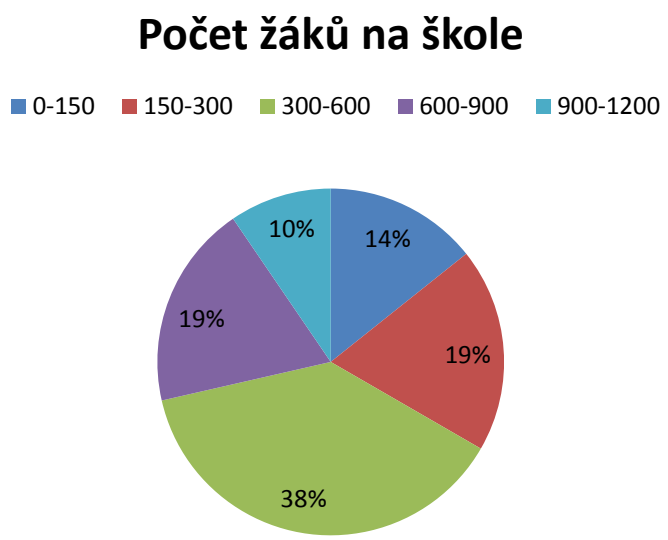
Číselně vyjádřeno, ze středních škol se zúčastnilo 20,5% oslovených institucí. Rozdělení středních škol bylo následné: 67% střední odborné školy, 16% odborných učilišť, 12% gymnázií a 5% speciálních škol. Zaměření je 41% technické, 31% odborné oborové, 19% všeobecné, 5% zdravotní, 2% umělecké a 2% humanitní. Velikost škol dle žáků bylo 71% škol o velikosti do šesti set žáků, zbytek byly školy do tisíc dvě stě žáků. Vybavení školy hodnotí pedagogové jako dobré a dostatečné. Učebna výpočetní techniky, po stránce vybavení, je využívána průměrně na 70%. Šedesát dva procent dotázaných učitelů očekává zlepšení vybavenosti školy v následujících dvou letech. Projektová výuka je užívána 45% dotázaných učitelů, průměrně je této formě výuky věnováno 26% z běžné výuky. Na středních školách má časový prostor na projektovou výuku 74% vyučujících. Olympiád a

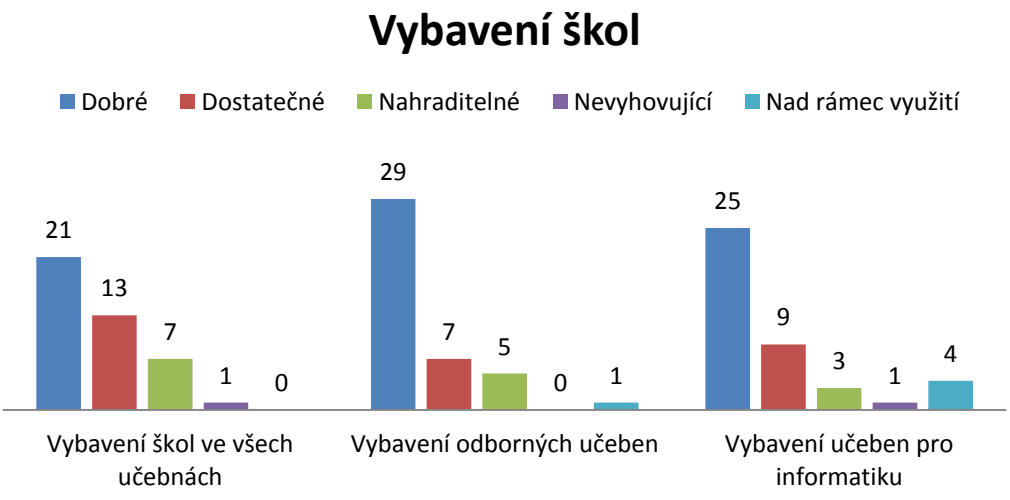
dalších různých soutěží v oblasti informatiky a tvorby SW se účastní žáci od 33% dotázaných pedagogů. Dle délky tvorby realizuje 47% dotázaných zejména krátkodobé projekty, 26% realizuje projekty v rámci dnů a 11% realizuje mimořádně dlouhodobé projekty. Učitelé pozorují, že žáci jsou více motivovaní, aktivní a tvořiví, oproti běžné výuce. Dále pozorují, že žáci jsou stejně pohotoví, vytrvalí, tolerantní a sebekritičtí jako v běžné výuce. Tvorbu SW aplikací uvádí 21% dotázaných, z toho 89% uvádí, že tvoří jednu až čtyři aplikace za školní rok a 11% tvoří více než sedm aplikací. Práce na projektech probíhá v 78% individuálně, skupinovou práci využívá 22% dotázaných učitelů. Dle rozvržení průměrné práce na projektu a jejím grafickým zobrazením je pozorovatelná podobnost s metodikou unifikovaného procesu (grafické hodnocení Obrázek 14).

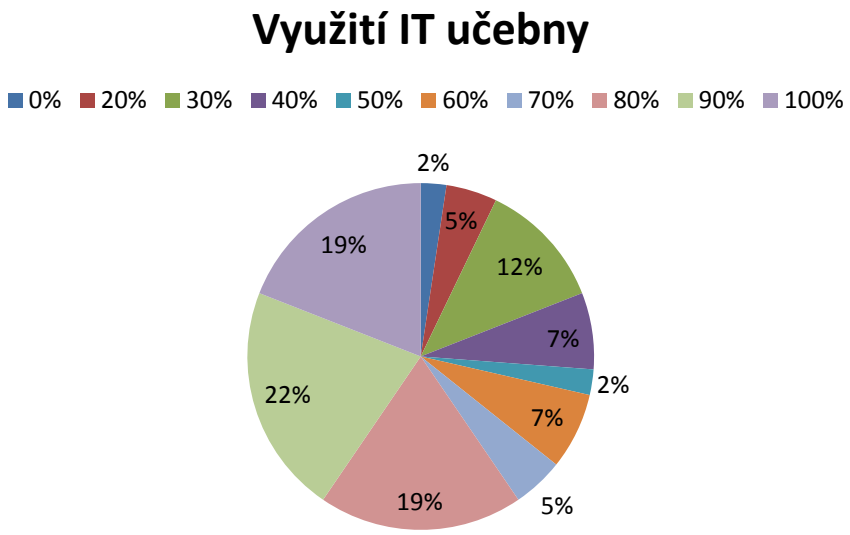


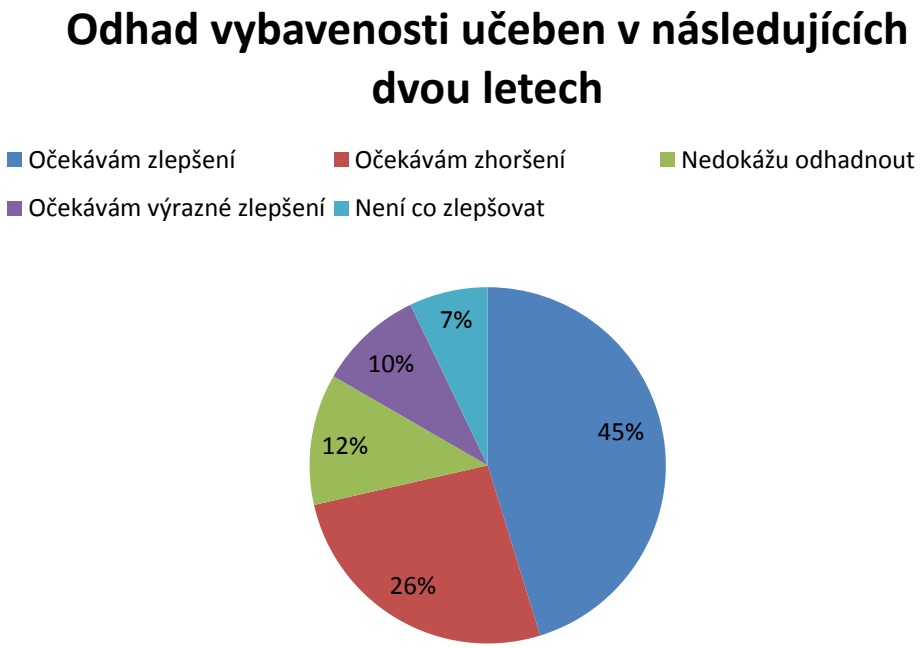


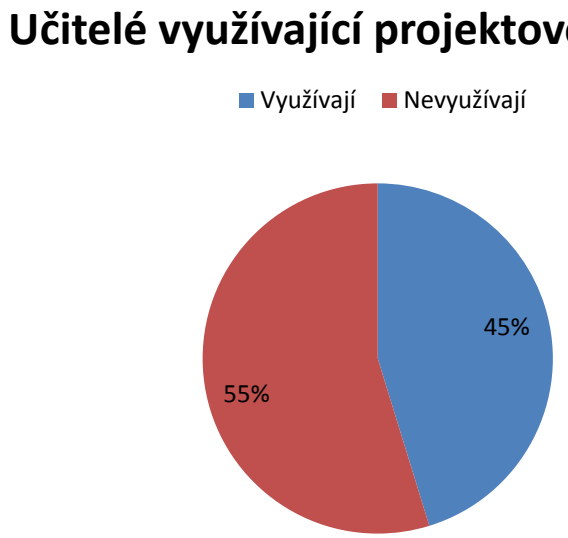

Zaměření středních škol
Technické
Odborné oborové
Obecné
Zdravotnické
Humanitní
Umělecké
2%
2%
5%
19%
41%
31%

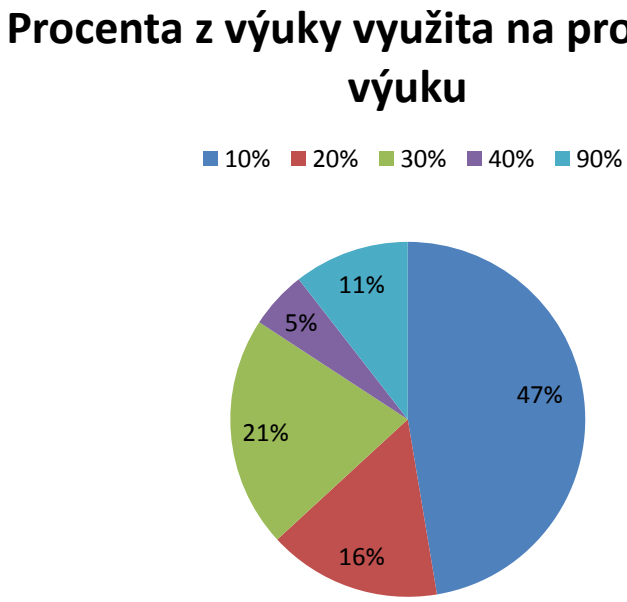

Počet žáků na škole
0-150
150-300
300-600
600-900
900-1200
10%
14%
19%
19%
38%

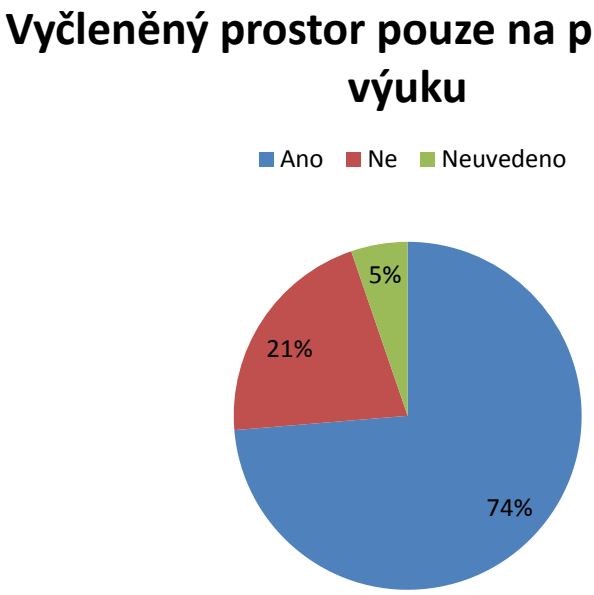

Vybavení škol
Dobré
Dostatečné
Nahraditelné
Nevyhovující
Nad rámec využití
21
13
7
1
0
29
7
5
0
1
25
9
3
1
4
Vybavení škol ve všech učebnách
Vybavení odborných učeben
Vybavení učeben pro informatiku

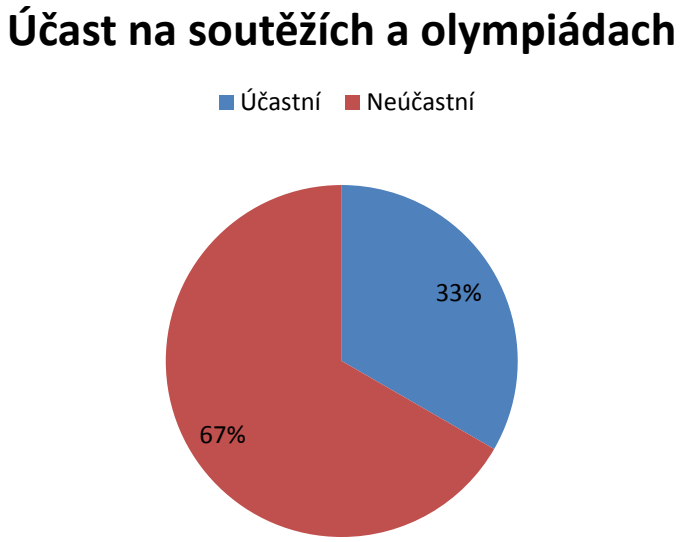

Využití IT učebny
0%
20%
30%
40%
50%
60%
70%
80%
90%
100%
2%
5%
12%
7%
2%
7%
5%
19%
22%
19%

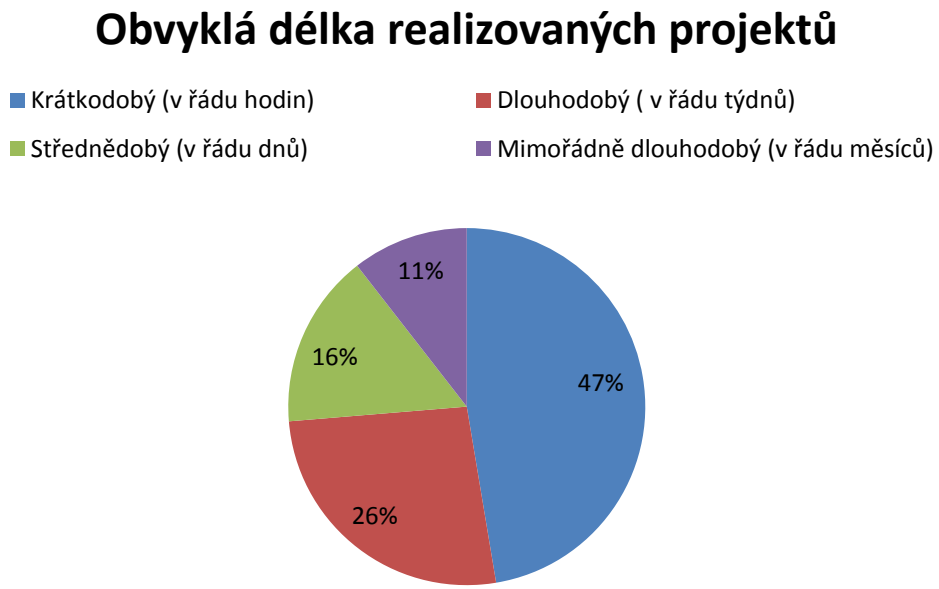

Odhad vybavenosti učeben v následujících dvou letech
Očekávám zlepšení
Očekávám zhoršení
Nedokážu odhadnout
Očekávám výrazné zlepšení
Není co zlepšovat
7%
10%
45%
12%
26%

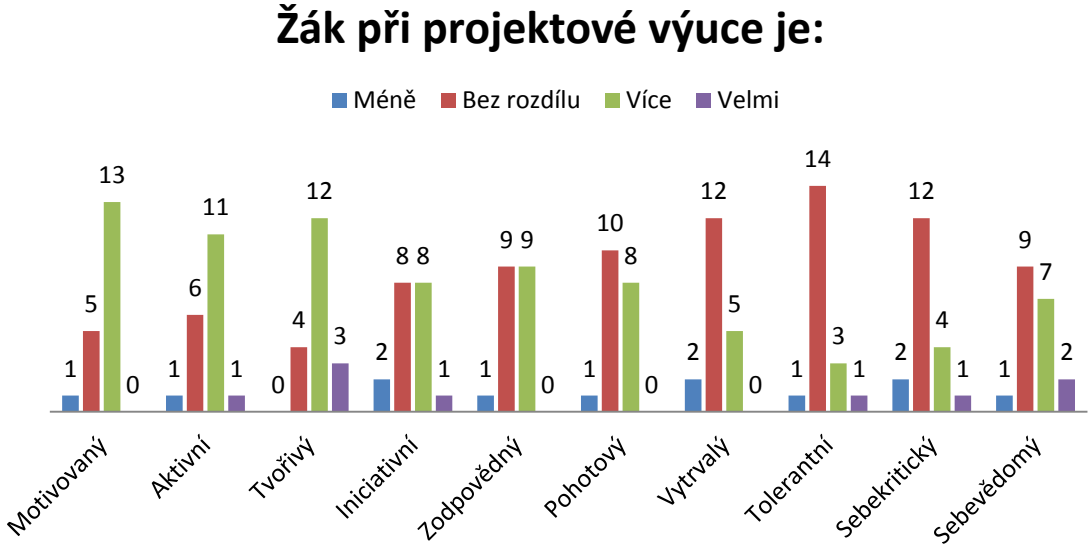

Učitelé využívající projektovou výuku
Využívají
Nevyužívají
45%
55%

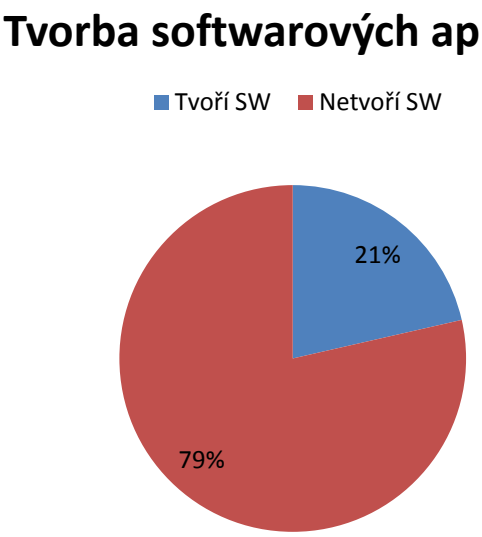

Procenta z výuky využita na projektovou výuku
10%
20%
30%
40%
90%
11%
5%
47%
21%
16%

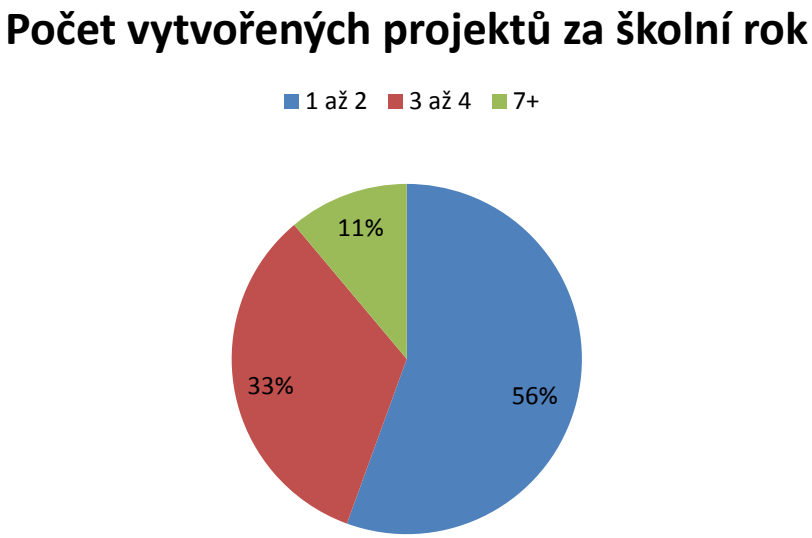

Vyčleněný prostor pouze na projektovou výuku
Ano
Ne
Neuvedeno
5%
21%
74%

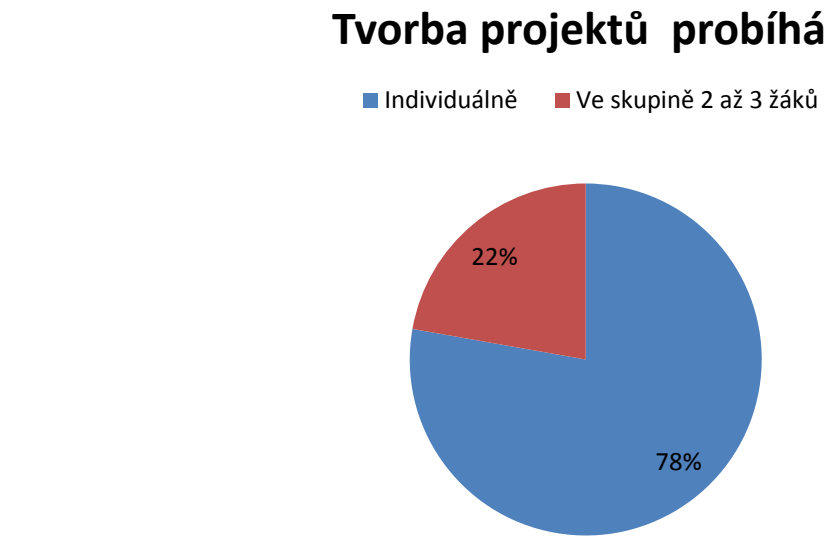

Účast na soutěžích a olympiádach
Účastní
Neúčastní
33%
67%

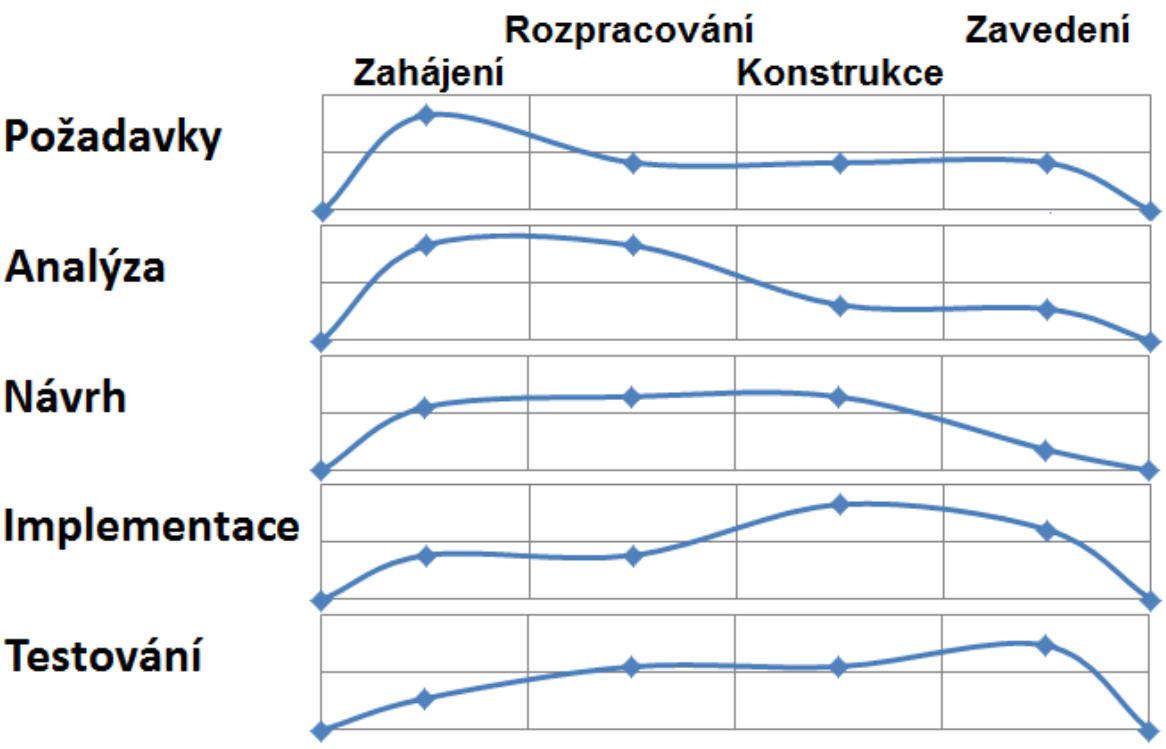

Obvyklá délka realizovaných projektů
Krátkodobý (v řádu hodin)
Dlouhodobý (v řádu týdnů)
Střednědobý (v řádu dnů)
Mimořádně dlouhodobý (v řádu měsíců)
11%
16%
47%
26%


Žák při projektové výuce je:
Méně
Bez rozdílu
Více
Velmi
1 5 13
1 6 11
0 4 12
2 8 8
1 9 9
1 10 8
2 12 5
1 14 3
2 12 4
1 9 7
0 1
1 0
3 1
1 0
0 1
0 1
1 2
Motivovaný
Aktivní
Tvořivý
Iniciativní
Zodpovědný
Pohotový
Vytrvalý
Tolerantní
Sebekritický
Sebevědomý


Tvorba softwarových aplikací
Tvoří SW
Netvoří SW
21%
79%


Počet vytvořených projektů za školní rok
1 až 2
3 až 4
7+
11%
33%
56%


Tvorba projektů probíhá:
Individuálně
Ve skupině 2 až 3 žáků
22%
78%


Průměrné rozvržení práce na středních školách
Zahájení
Rozpracování
Konstrukce
Zavedení
Požadavky
Analýza
Návrh
Implementace
Testování

6 NÁVRH METODICKÉHO POSTUPU

Návrh metodiky probíhal na základě vyhodnoceného dotazníku. Na základě vyhodnocení byla zvolena jako nejbližší metodika unifikovaného procesu pro vývoj aplikací. Tato metodika je nejlépe použitelná pro střední školy, ve kterých probíhá projektová výuka a jejichž žáci tvoří softwarové produkty. Hlavním důvodem je rigoróznost této metodiky, stejně tak i ve škole je nutné přesné a jasné zadání na počátku práce. Neočekávají se další změny v průběhu tvorby jako u agilních metodik. V úvodní části metodiky jsou vybrány hlavní výstupy, na které by se mělo zaměřit. Tyto výstupy jsou přebrány z rámcového vzdělávacího plánu (RVP). Protože RVP pro střední školy je specifický pro každý obor, je využit vzdělávací plán pro gymnázia, který by dle obecných předpokladů měl naplňovat všeobecné středoškolské vzdělání, a tím připravit žáky k dalšímu studiu. Metodika má za cíl vytvořit určitý postup projektem s prvky unifikovaného procesu. Proto je rozdělena na fáze, stejně jako metodika UP. V každé fázi je na počátku obecně popsáno, co je její náplní. Pokud je vhodná iterace v dané fázi, je uveden seznam částí, ve kterých jsou jasně dané úkoly, kdy v každé fázi probíhá iterativní opakování různých částí do té doby, než jsou splněny cíle dané fáze. Tímto způsobem práce se eliminují chyby týkající se zvládnutí zadání. Výhodou je vyčlenění problému a práce na něm z obecného řešení do konkrétní podoby a chyby, které se mohou vyskytnout, lze zjistit a odstranit na začátku. Navíc si žáci mohou vyzkoušet, jakým způsobem probíhá tvorba SW v praxi. Je nutné podotknout, že i když se organizace řídí nějakou metodikou, ne vždy ji plně dodržována. Pro celkovou lepší orientaci je návrh metodiky doplněn obrázky, které zvýrazňují části pro učitele, žáka a příklad.(Obrázek 19) Pro příklad byl použit projekt, který je možné realizovat tak, aby obsahoval co nejvíce prvků projektové výuky, tak i plnil požadavky z RVP a při tvorbě bylo možné uplatnit navrženou metodiku práce.

Obrázek 19 - Ikony pro usnadnění orientace - učitel, žák a příklad.

7 KONFRONTACE S PRAXÍ NA ZŠ A SŠ

Vytvořené metodiky byly konzultovány s učiteli na základní a střední škole, kde jsem během magisterského studia absolvoval průběžnou praxi. Obě metodiky byly konzultovány formou rozhovoru na dané téma. Otázky byly – zda je výsledná práce vhodná pro uplatnění v praxi, popřípadě jakým způsobem, kde vidí učitelé nedostatky nebo přednosti, zda by ji využili ve vlastní výuce. Na základní škole jsme dospěli k následujícímu. Žáci mají jednou týdenně hodinu informatiky, a to pouze na druhém stupni. Pomůcky, jako jsou učebnice do informatiky, škola nevyužívá. Standardně si žáci vedou elektronický sešit. Učebna pro informatiku je menší, obsahuje sedmnáct počítačů, proto je nutné i třídu rozdělit na dvě poloviny. Vyučující hodnotil metodiku jako „těžce“ použitelnou za současného stavu. Žáci jsou vedeni podobným způsobem, nicméně samostatná práce by zabrala velký čas navíc, který není. Jako výhody byly označeny nové možnosti, které je možné aplikovat, jiný pohled na projektovou výuku. Co se týká samotné práce, některé body by mohly být vynechány, jako například domluva s žáky a jejich schválení zadání a časového harmonogramu. Příklady uvedené by mohly být podrobnější, mohly by obsahovat šablony, které by učitelé mohli doplňovat. Při následném hodnocení celkový přehled, co konkrétně a jak hodnotit. Na závěr by mohly být uvedeny další příklady. Ve výuce použita nebude, spíš v přípravě na výuku jako možná alternativa. Tento způsob výuky by někteří žáci nezvládli, proto bude dále volena forma živého návodu (učitel předvádí na projektoru a žáci současně plní zadaný úkol, k tomu si vytváří vlastní poznámky). Při konzultaci na střední škole byla použita metodika pro střední školy. Vhodnost uplatnění pro praxi byla definována jakou použitelná pro začínající učitele, ovšem pouze jako návrh na vedení projektů, pro začínající učitele nebo pro ty, kteří hledají způsob. Metodika byla hodnocena jako obtížnější pro práci žáků. Z toho důvodu, že je, si sami musí určit velikost a část iterace, kterou v dané fázi provedou. Navíc nejdříve by bylo nutné porozumět samotné metodice práce a až následně tvořit, programovat. Chybí podrobnější popis iterací, jakým způsobem rozdělit danou práci, kolik iterací je vhodných. Naopak velkou výhodu je příprava žáků do praxe, pokud se ovšem v praxi s tímto způsobem potkají. Další otázkou, která byla a je aktuální, zda je vhodné využití programovacího jazyka Pascal, s touto navrženou metodikou (pro jednoduchost a názornost pochopení základů programování je na škole zaveden programovací jazyk Pascal). Ve výuce je použitelná, ale je nutné začít pracovat s žáky, kteří začínají programovat, těžko by se začlenila na konci ročníku.

ZÁVĚR

Při tvorbě diplomové práce jsem provedl rešerši metodik používaných v praxi. Následně jsem prováděl výzkum formou dotazníku, kterého se celkově zúčastnilo 13,75% dotázaných. I přes to, že celkový počet byl malý, získal jsem přehled o spokojenosti učitelů s vybavením škol, jak moc je využito vybavení učebny pro výpočetní techniku. Důležité bylo zjištění, jak moc se učitelé věnují projektové výuce. Protože pokud chtějí aplikovat navrženou metodiku, je vhodné přiblížit tomu i celkový způsob práce. Jako nejvhodnější a zároveň i nejběžněji používanou jsem zvolil metodiku unifikovaného procesu vývoje. Ta je mohutně podporovaná nástroji unifikovaného modelovacího jazyka. V navržené metodice v příkladu prezentuji jeden statický model systému a dynamický, který není tak běžný. Stejně jako v běžné praxi zdůrazňuji tvorbu kvalitní dokumentace, ze které je možné vycházet v následujících aplikacích. V závěru návrhu připomínám vytvoření vlastních návrhových vzorů, které nemusí být tak propracované, ale po prostudování by měly nasměrovat žáka správným směrem při další tvůrčí činnosti. Důležitým je i faktor motivace, jenž může přispět k sebeprezentaci žáka a zároveň pomoct pedagogovi. Například vytvoření si složky žáka, kterou si vede samotný žák se svými výtvory a následně si můžou rodiče na třídních schůzkách sami ověřit danou práci, popřípadě porovnat s ostatními žáky. Navrhovaná metodika zahrnuje výsledné požadavky kladené na žáka RVP. Na závěr jsem jednotlivé metodiky konzultoval na základní a střední škole, kde jsem prováděl během studia průběžnou praxi. V oddílu „Konfrontace s praxí na ZŠ a SŠ“ jsou shrnuty připomínky a doplňky, které mají za úkol celou metodiku přiblížit do praxe a tím i přispět k lepší přípravě žáků.

ZÁVĚR V ANGLIČTINĚ

In creating diploma thesis I made a search of methodologies used in practice. Subsequently, I performed questionnaire research, which participated total 13.75% of respondents. Despite the fact that the total number was small, I gained an overview of teachers' satisfaction with the equipment of schools, how much is equipment exploited in classroom computing. An important finding was how much teachers use project education. Because if you want to apply the proposed methodology, it is advisable to bring the overall way of working were adapted. The best and also most commonly used method I chose the unified process development. This is powerfully supported by the unified modeling language tools. In the proposed methodology in the example I present a model of static and dynamic system that is not so common. As in normal practice, emphasizing the creation of quality documentation, from which you can build on these applications. In conclusion, I note the proposal to create your own design patterns, which may not be so elaborate, but the study should direct the pupil in the right direction in other creative activities. An important factor is motivation, which can contribute to student self-presentation and also help the lecturer. The proposed methodology includes the resulting demands placed on the student's RVP (FEP). Finally I consulted the methodologies on primary and secondary school, on which I performed continual teaching practice. In the section "Konfrontace s praxí na ZŠ a SŠ" comments and supplements are summarized and have to keep the methodology closer into practice and hence contribute to better prepare students.

SEZNAM POUŽITÉ LITERATURY

[1] GUCKENHEIMER Sam, PEREZ Juan J.. *Efektivní softwarové projekty*. Vyd. 1. Překlad Jan Kuklínek, Jan Pokorný. Brno: Zoner Press, 2007, 255 s. ISBN 978-808-6815-626.

[2] BECK Kent. *Extrémní programování*. Vyd. 1. Praha: Grada, 2002, 158 s. ISBN 80-247-0300-9.

[3] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací*: *objektově orientovaná analýza a návrh prakticky*. Vyd. 1. Překlad Bogdan Kiszka. Brno: Computer Press, 2007, 567 s. ISBN 978-802-5115-039.

[4] GAMMA Erich, HELM Richard, JOHNSON Ralph, VLISSIDES John. *Návrh programů pomocí vzorů*: *Stavební kameny objektově orientovaných programů*. Praha: Grada Publishing, 2003. ISBN 80-247-0302-5.

[5] MEŠKO Dušan, KATUŠČÁK Dušan. *Akademická príručka*. 1. slovenské vyd. Martin: Osveta, c2004, 316 s. ISBN 80-806-3150-6.

[6] *Rámcový vzdělávací program pro základní vzdělávání.* [online]. Praha: Výzkumný ústav pedagogický v Praze, 2007. 126 s. [cit. 2012-03-12]. Dostupné z WWW:<http://www.vuppraha.cz/wp-content/uploads/2009/12/RVPZV_2007-07.pdf>.

[7] *Rámcový vzdělávací program pro gymnázia.* [online]. Praha: Výzkumný ústav pedagogický v Praze, 2007. 100 s. [cit. 2012-03-12]. Dostupné z WWW: <http://www.vuppraha.cz/wp-content/uploads/2009/12/RVPG-2007-07_final.pdf>. ISBN 978-80-87000-11-3.

[8] BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů*: *kategorizace, agilní metodiky, vzory pro návrh metodiky*. 1. vyd. Praha: Grada, 2005, 163 s. ISBN 80-247-1075-7.

[9] KOLÁŘ, Petr. Operační systémy. In: *Operační systémy* [online]. 2005 [cit. 2012-03-15]. Dostupné z: http://www.kai.vslib.cz/~kolar/os/

[10] KŘENA, Bohuslav a Radek KOČÍ. Úvod do softwarového inženýrství. In: *Http://www.scribd.com* [online]. 2006 [cit. 2012-03-15]. Dostupné z: http://www.scribd.com/doc/48636337/5/Historie-softwaroveho-in%C5%BEen%C3%BDrstvi

[11] DVOŘÁK, Miloš. *Návrhové vzory (design paterns)* [online]. 2008 [cit. 2012-03-17]. Dostupné z: http://objekty.vse.cz/Objekty/Vzory

[12] KRAVAL, Ilja. *Objektové modelování pomocí UML v praxi, 2005* [online]. ©2005 [cit. 2012-03-17]. Dostupné z: http://www.objects.cz/

[13] Kučerová, Helena. Metodiky ontologického inženýrství. *Ikaros* [online]. 2011, roč. 15, č. 5 [cit. 2012-03-17]. Dostupné z: http://www.ikaros.cz/node/6801. URN-NBN:cz-ik6801. ISSN 1212-5075.

[14] TutorialsPoint: Simply Easy Learning. *© 2012 TutorialsPoint.COM* [online]. [cit. 2012-03-17]. Dostupné z: http://www.tutorialspoint.com/uml/

[15] FREEMAN, Eric a Elisabeth FREEMAN. *Head first design patterns*. 1st ed. Sebastopol: O´Reilly, 2004, 638 s. ISBN 05-960-0712-4.

[16] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003, 219 s. ISBN 80-731-5039-5.

[17] VALIŠOVÁ, Alena a Hana KASÍKOVÁ. *Pedagogika pro učitele*. Vyd. 1. Praha: Grada, 2007, 402 s. Pedagogika (Grada). ISBN 978-802-4717-340.

[18] CHRÁSKA, Miroslav. *Metody pedagogického výzkumu*: *základy kvantitativního výzkumu*. Vydání 1. Praha: Grada Publishing, 2007, 265 s. ISBN 978-80-247-1369-4.

[19] GOOGLE INC. *Google* [online]. 2011 [cit. 2012-03-24]. Dostupné z: http://www.google.cz/

[20] Mozilla. MOZILLA FOUNDATION. *Mozilla firefox* [online]. 2012 [cit. 2012-04-09]. Dostupné z: http://www.mozilla.org/en-US/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ALGOL	ALGOrithmic Language - programovací jazyk
ASD	Adaptive Software Development - metodika adaptativního SW vývoje
COBOL	COmmon Business Oriented Language - programovací jazyk
DSDM	Dynamic Systém Development Method
ERD	Entity Relationship Diagram - vzájemný vztah mezi entitami
FDD	Featrue Driven Developement - metodika řízena užitnými vlastnostmi
FORTRAN	FORmula TRANslation - programovací jazyk
GoF	Gang of Four - zakladatelé myšlenky návrhových vzorů
HW	Hardware
IT	Informační Technologie
METES	Methodology Evaluation System - Systém vyhodnocování metodik
MSDN	Microsoft Developer Network
OMG	Object Management Group - sdružení firem zabývající se IT
OML	OPEN Modeling Language - nástroj pro pohledy v metodice OPEN
OPF	OPEN Process Framework - rámce metodiky OPEN
RVP / FEP	Rámcový Vzdělávací Plán / Federal Education Plan
SŠ	Střední Škola
SW	Software
UML	Unified Modeling Language - metodika návrhu SW
UP	Unified Process
VSTS	Visual Studio Team System
ZŠ	Základní Škola

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: DOTAZNÍK

Tvorba softwarového projektu na ZŠ a SŠ.

Vítám Vás při vyplnění dotazníku.

Tento dotazník a zejména jeho výsledky budou sloužit pro diplomovou práci "Tvorba softwarového projektu - metodická pomůcka pro základní a střední školy". Je určen především učitelům informatiky a vyučujícím s tímto oborem spojených.

Rozsah dotazníku: 6 stran, 22 otázek.
Časová náročnost: 7 - 15 minut.

(některé otázky jsou struktorovanější, jiné nemusí být zobrazeny vůbec - zobrazení dle předchozích odpovědí)

Předem děkuji.
Bc. Marek Zmeškal.
Dotazy: m_zmeskal@fai.utb.cz
*Povinné pole

Identifikace školy

1. Typ školy ? *

- ○ Základní Škola
- ○ Odborné Učiliště
- ○ Střední Odborná Škola
- ○ Gymnázium
- ○ Speciální Škola
- ○ Jiné:

2. Zaměření školy ? *

- ○ Technické
- ○ Humanitní
- ○ Odborné oborové
- ○ Obecné
- ○ Umělecké
- ○ Jiné:

3. Počet žáků na škole? *

- ○ Do 150
- ○ 150-300
- ○ 300-600
- ○ 600-900
- ○ 900-1200
- ○ 1200 a více

Vybavení školy

4. Doplňte dle vlastních zkušeností : *

	Nevyhovující	Nahraditelné	Dostatečné	Dobré	Nad rámec využití
Obecné vybavení školy - ve všech učebnách	○	○	○	○	○
Vybavení odborných učeben	○	○	○	○	○
Vybavení učeben pro Informatiku	○	○	○	○	○

5. Z jaké části se využívá vybavení IT učeben ? *

0 %

6. Co očekáváte ve vybavenosti učeben, v období příštích dvou let ? *

- ○ Očekávám zhoršení
- ○ V tomto směru není co zlepšovat
- ○ Nedokážu odhadnout
- ○ Očekávám zlepšení
- ○ Očekávám výrazné zlepšení

Pokračovat »

Tvorba softwarového projektu na ZŠ a SŠ.

*Povinné pole

Projektová výuka

7. Využíváte ve výuce "Projektovou Výuku" ? *

- ○ Ano
- ○ Ne

« Zpět | Pokračovat »

Tvorba softwarového projektu na ZŠ a SŠ.

Projektová výuka pokračování

8. Z kolika procent výuky, využíváte právě projektovou výuku ?

10 %

9. Co si představujete pod pojmem projektová výuka ?

10. Je vyčleněn časový prostor přímo pro praktické využití znalostí ?
(praktické činnost, měření v terénu, praxe, programování, CAD/CAM, ...)

- Ano
- Ne

11. Účastní se žáci soutěží, olympiád zaměřených na tvorbu softwaru ?

- Ano
- Ne

12. Jaký rozsah projektové výuky volíte nejčastěji?

Krátkodobý (v řádu hodin)

13. Doplňte dle zkušeností, chování žáků při PROJEKTOVÉ výuce v porovnání s běžnou výukou :

Žák je

	méně	bez rozdílu	více	velmi
motivovaný	○	○	○	○
aktivní	○	○	○	○
tvořivý	○	○	○	○
iniciativní	○	○	○	○
zodpovědný	○	○	○	○
pohotový	○	○	○	○
vytrvalý	○	○	○	○
tolerantní	○	○	○	○
sebekritický	○	○	○	○
sebevědomý	○	○	○	○

14. Kde žáci prezentují svou práci tvořenou prostřednictvím projektové výuky ?

« Zpět | Pokračovat »

Tvorba softwarového projektu na ZŠ a SŠ.

*Povinné pole

Vývoj / Tvorba softwaru

15. Tvoří žáci nové softwarové produkty ? *

- Ano
- Ne

« Zpět | Pokračovat »

Používá technologii Dokumenty Google

Tvorba softwarového projektu na ZŠ a SŠ.

Vývoj / Tvorba softwaru pokračování

16. Kolik softwarových produktů vytvoří za školní rok ?

1 - 2

17. Tvoří softwarové produkty ve skupinách ?

Individuálně

18. Kolik času z projektu věnují žáci následujícím úkolům ve fázi ZAHÁJENÍ ?
(Zahájení - definování cílů projektu, požadavky, sestavení harmonogramu, vytvoření modelu, vytvoření jednoduchého prototypu)

	0	25	50	75	100 [%]
Požadavkům					
Analýze					
Návrhu					
Implementace					
Testování					

19. Kolik času z projektu věnují žáci následujícím úkolům ve fázi ROZPRACOVÁNÍ ?
(Rozpracování - definování architekturu systému, vytvoření protctypu pro ověření architektonických principů, umožnění upřesnění plánu realizace, definice znovupoužitelných komponent)

	0	25	50	75	100 [%]
Požadavkům	○	○	○	○	○
Analýze	○	○	○	○	○
Návrhu	○	○	○	○	○
Implementace	○	○	○	○	○
Testování	○	○	○	○	○

20. Kolik času z projektu věnují žáci následujícím úkolům ve fázi KONSTRUKCE ?
(Konstrukce - návrh, realizace systému, testování)

	0	25	50	75	100 [%]
Požadavkům	○	○	○	○	○
Analýze	○	○	○	○	○
Návrhu	○	○	○	○	○
Implementace	○	○	○	○	○
Testování	○	○	○	○	○

21. Kolik času z projektu věnují žáci následujícím úkolům ve fázi ZAVEDENÍ ?
(Zavedení - zajištění použitelnosti SW, školení uživatelů, předání dokumentace, vytvoření podpory, prezentace)

	0	25	50	75	100 [%]
Požadavkům	○	○	○	○	○
Analýze	○	○	○	○	○
Návrhu	○	○	○	○	○
Implementace	○	○	○	○	○
Testování	○	○	○	○	○

« Zpět | Pokračovat »

Tvorba softwarového projektu na ZŠ a SŠ.

*Povinné pole

Doplňkové údaje

Vyhodnocení : *

- [ ] Mám zájem o výsledky dotazníku.
- [ ] Mám zájem o výslednou praktickou práci (příloha).
- [ ] Mám zájem o teoretickou práci (dokument).

Další kontakt : *

- ( ) Můžu být kontaktován pro další konzultaci - vyplním kontaktní údaje.
- ( ) Nepřeji si být dále kontaktován.

Kontakt

Jméno , příjmení:

E-mail:

Telefonní číslo:

« Zpět | Odeslat

Tvorba softwarového projektu na ZŠ a SŠ.

Vítám Vás při vyplnění dotazníku.

Pro vyplnění si zvolte předmět a obor, který se nejvíc zabývá informační technologií. Dotazník můžete vyplnit i více krát pro různý obor a předmět.

Tento dotazník a zejména jeho výsledky budou sloužit pro diplomovou práci "Tvorba softwarového projektu - metodická pomůcka pro základní a střední školy". Je určen především učitelům informatiky a vyučujícím s tímto oborem spojených.

Rozsah dotazníku: 6 stran, 22 otázek.
Časová náročnost: 7 - 15 minut.
(některé otázky jsou víc strukturované, jiné nemusí být zobrazeny vůbec - zobrazení dle předchozích odpovědí)

Předem děkuji.
Bc. Marek Zmeškal.
Dotazy: m_zmeskal@fai.utb.cz

*Povinné pole

Identifikace školy

1. Typ školy? *

- Základní Škola
- Odborné Učiliště
- Střední Odborná Škola
- Gymnázium
- Speciální Škola
- Jiné:

2. Zaměření školy? *

- Technické
- Humanitní
- Odborné oborové
- Obecné
- Umělecké
- Jiné:

3. Počet žáků na škole? *

- Do 150
- 150-300
- 300-600
- 600-900
- 900-1200
- 1200 a více

Vybavení školy

4. Doplňte dle vlastních zkušeností: *

	Nevyhovující	Nahraditelné	Dostatečné	Dobré	Nad rámec využití
Obecné vybavení školy - ve všech učebnách					
Vybavení odborných učeben					

	Nevyhovující	Nahraditelné	Dostatečné	Dobré	Nad rámec využití
Vybavení učeben pro Informatiku					

5. Z jaké části se využívá vybavení IT učeben? *
(využití zařízení jako je: interaktivní tabule, multimediální tabule, PC, projektor, měřící zařízení ... 0% = tabule + křída)

- 0 %
- 10 %
- 20 %
- 30 %
- 40 %
- 50 %
- 60 %
- 70 %
- 80 %
- 90 %
- 100 %

6. Co očekáváte ve vybavenosti učeben, v období příštích dvou let? *

- Očekávám zhoršení
- V tomto směru není co zlepšovat
- Nedokážu odhadnout
- Očekávám zlepšení

- Očekávám výrazné zlepšení

Projektová výuka

7. Využíváte ve výuce "Projektovou Výuku"? *

- Ano
- Ne (šedou část není nutné vyplnit)

Projektová výuka pokračování

8. Z kolika procent výuky, využíváte právě projektovou výuku ?

- 10 %
- 20 %
- 30 %
- 40 %
- 50 %
- 60 %
- 70 %
- 90 %
- 100 %

9. Co si představujete pod pojmem „projektová výuka“?

10. Je vyčleněn časový prostor přímo pro praktické využití znalostí ? (praktické činnost, měření v terénu, praxe, programování, CAD/CAM, ...)

- Ano
- Ne

11. Účastní se žáci soutěží, olympiád zaměřených na tvorbu softwaru ?

- Ano
- Ne

12. Jaký rozsah projektové výuky volíte nejčastěji?

- Krátkodobý (v řádu hodin)
- Střednědobý (v řádu dnů)
- Dlouhodobý (v řádu týdnů)
- Mimořádně dlouhodobý (v řádu měsíců, realizováno i mimo výuku)

13. Doplňte dle zkušeností, chování žáků při PROJEKTOVÉ výuce v porovnání s běžnou výukou : Žák je

	méně	bez rozdílu	více	velmi
motivovaný				
aktivní				
tvořivý				
iniciativní				
zodpovědný				
pohotový				
vytrvalý				
tolerantní				

	méně	bez rozdílu	více	velmi
sebekritický	○	○	○	○
sebevědomý	○	○	○	○

14. Kde žáci prezentují svou práci tvořenou prostřednictvím projektové výuky ?

Vývoj / Tvorba softwaru

15. Tvoří žáci nové softwarové produkty? *
(programy, ve kterých se projeví tvůrčí činnost žáka, závěrečné projekty, …)

- ○ Ano
- ○ Ne (žlutou část není nutné vyplnit)

Vývoj / Tvorba softwaru pokračování

16. Kolik softwarových produktů vytvoří za školní rok?

- ○ 1 - 2
- ○ 3 - 4
- ○ 5 - 6
- ○ 7+

17. Tvoří softwarové produkty ve skupinách ?

- ○ Individuálně
- ○ 2 - 3 žáci
- ○ 4 - 5 žáků

- [ ] 6 a více žáků

18. Kolik času z projektu věnují žáci následujícím úkolům ve fázi ZAHÁJENÍ? (z celkového projektu cca 0-20%)
(Zahájení - definování cílů projektu, požadavky, sestavení harmonogramu, vytvoření modelu, vytvoření jednoduchého prototypu)

	0	25	50	75	100 [%]
Požadavkům	○	○	○	○	○
Analýze	○	○	○	○	○
Návrhu	○	○	○	○	○
Implementace	○	○	○	○	○
Testování	○	○	○	○	○

19. Kolik času z projektu věnují žáci následujícím úkolům ve fázi ROZPRACOVÁNÍ? (z celkového projektu obvykle cca 5-30%)
(Rozpracování - definování architekturu systému, vytvoření prototypu pro ověření architektonických principů, umožnění upřesnění plánu realizace, definice znovupoužitelných komponent)

	0	25	50	75	100 [%]
Požadavkům	○	○	○	○	○
Analýze	○	○	○	○	○
Návrhu	○	○	○	○	○
Implementace	○	○	○	○	○
Testování	○	○	○	○	○

20. Kolik času z projektu věnují žáci následujícím úkolům ve fázi KONSTRUKCE? (z celkového projektu cca 30-90%)
(Konstrukce - návrh, realizace systému, testování)

	0	25	50	75	100 [%]
Požadavkům	○	○	○	○	○
Analýze	○	○	○	○	○
Návrhu	○	○	○	○	○
Implementace	○	○	○	○	○
Testování	○	○	○	○	○

21. Kolik času z projektu věnují žáci následujícím úkolům ve fázi ZAVEDENÍ? (z celkového projektu cca 0-20%)
(Zavedení - zajištění použitelnosti SW, školení uživatelů, předání dokumentace, vytvoření podpory, prezentace)

	0	25	50	75	100 [%]
Požadavkům	○	○	○	○	○
Analýze	○	○	○	○	○
Návrhu	○	○	○	○	○
Implementace	○	○	○	○	○
Testování	○	○	○	○	○

Doplňkové údaje

Vyhodnocení: *

- ☐ Mám zájem o výsledky dotazníku.
- ☐ Mám zájem o výslednou praktickou práci (příloha).
- ☐ Mám zájem o teoretickou práci (dokument).

Další kontakt: *

- ○ Můžu být kontaktován pro další konzultaci - vyplním kontaktní údaje.
- ○ Nepřeji si být dále kontaktován.

Kontakt

Jméno, příjmení:

E-mail:
Telefonní číslo:

PŘÍLOHA P II: NÁVRH METODIKY PRO ZŠ

Metodická pomůcka pro tvorbu softwarových aplikací na základních školách.

Formou projektové výuky s prvky unifikovaného procesu vývoje softwarových aplikací.

Úvod

Metodická příručka je určena pedagogům základních škol. Její využití je především ve výuce informatiky, při práci se softwarovými aplikacemi. Hlavní postup práce je veden metodikou unifikovaného vývoje softwaru.

Vzhledem k RVP a náplní učiva základních škol se nepředpokládá výrazný podíl žáků na tvorbě softwarových aplikací. Cílem je rozšíření a upevnění dovedností žáků s prací běžnými počítačovými produkty. Dosažení rozhledu s využitím internetu a online aplikacemi i individuální tvorby prezentace a reálného posouzení vlastní tvůrčí práce.

Metodika unifikovaného procesu je přiblížena prostředí základní školy a požadavkům, není vázaná na konkrétní aplikace.

Obsah

Výstupní požadavky dle RVP

Vybrané požadavky, které jsou pro danou metodiku důležité.

Žák (1. stupeň):

- využívá základní standardní funkce počítače a jeho nejběžnější periferie
- respektuje pravidla bezpečné práce s hardware i software a postupuje poučeně v případě jejich závady
- chrání data před poškozením, ztrátou a zneužitím
- při vyhledávání informací na internetu používá jednoduché a vhodné cesty
- vyhledává informace na portálech, v knihovnách a databázích
- komunikuje pomocí internetu či jiných běžných komunikačních zařízení
- pracuje s textem a obrázkem v textovém a grafickém editoru

Žák (2. stupeň):

- ověřuje věrohodnost informací a informačních zdrojů, posuzuje jejich závažnost a vzájemnou návaznost
- ovládá práci s textovými a grafickými editory i tabulkovými editory a využívá vhodných aplikací
- uplatňuje základní estetická a typografická pravidla pro práci s textem a obrazem
- pracuje s informacemi v souladu se zákony o duševním vlastnictví
- používá informace z různých informačních zdrojů a vyhodnocuje jednoduché vztahy mezi údaji
- zpracuje a prezentuje na uživatelské úrovni informace v textové, grafické a multimediální formě

Tvorba softwarového produktu

Metodická pomůcka je rozdělena na několik částí, které jsou použity v případě tvorby softwarových produktů. Fáze na sebe vzájemně a plynule navazují, přičemž u každé je jasně daný cíl. Dále jsou uvedeny úkoly, které by měly být plněny především žáky i pedagogy. Jednotlivé části jsou koncipovány tak, aby zpřehlednily a ujasnily práci na daném projektu a přinesly především žákům přehled a základní informace o postupu práce při práci a tvorbě softwarového produktu. Před započetím je nutné, aby žáci měli dostatečné teoretické znalosti nebo bylo možné tyto znalosti získat během práce na jejich díle.

Role jsou dány přirozenou pozicí, která je obvyklá pro běžnou výuku v České republice. Úkoly jsou dány jak z pohledu práce pedagoga, tak souborem cílů práce žáků.

Jelikož metodika vychází z procesu vývoje aplikací a na základní škole, dle RVP pro základní školy, není tvorba softwarových produktů požadována (implementace zdrojového kódu, tvorba samostatně spustitelných aplikací). Bude řešen způsob práce s určitým problémem. Hlavním přínosem má být postupné řešení situace algoritmickým způsobem. Žáci by měli získat poznatek, jak situaci analyzovat, navrhnout řešení tak, aby jej byli schopni řešit individuálně. Při řešení složitějších úkolů rozložení na dílčí úkoly. Schopnosti efektivně hledat informace a použít je v praxi. To vše pod odborným dohledem, který pouze žáky směruje k cíli – využití výsledku dále při studiu nebo v praxi.

Předpokladem samostatné nebo skupinové práce je předchozí ukázka a procvičení dílčích úkolů. Tímto projektem by měli žáci své znalosti propojit a zopakovat si je.

Zahájení

Je to počáteční fáze, jedná se o odstartování celého projektu. Seznámení s tématem a vytvoření osobní motivace pro konečné použití. Případné rozdělení do týmů, následné shrnutí nových poznatků, které budou prohloubeny.

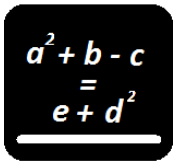
- ***Pedagog***

1. sdělí požadavky na projekt – zadání.
2. informuje o tom, kdo, kdy a jak bude projekt využívat.
3. návrh pracovního rozvrhu.
4. rozebrání možných rizik.
5. návrh řešení problémových částí.

- ***Žáci***

1. navrhují, upravují požadavky na projekt.
2. schválení pracovního rozvrhu.
3. pochopení klíčových momentů.
4. tvorba počátečních dokumentů pro dokumentaci.

- ***Příklad***



Pedagog:
Zadání:

Projekt se nazývá „Elektronický sešit“. V první části si vyzkoušíte navrhnout klasickou složkovou strukturu, v programu „Průzkumník“. Sloužit to bude především vám. Ve druhé části se pokusíte nalézt možnosti, na jakých stránkách lze vytvořit něco podobného, aby se ke složkám a dokumentům v nich mohlo přistupovat online. Je to výhodné hlavně pro vás, tím že máte vše online, nemůže se stát, že si zapomenete například referát doma. O přestávce máte možnost si jej vytisknout v knihovně nebo v počítačové učebně. Vše řádně zdokumentujte, využijte „Print Screen“. Pokud budete kopírovat obrázky nebo používat texty, uveďte zdroje do seznamu zdrojů. Šikovnější můžou využít internet na to, aby jim správné citace vytvořil. Vyhrazený čas jsou tři vyučovací hodiny plus jedna vyučovací hodina na prezentaci. Nejlepší návrh a popis budou zveřejněny na nástěnce nebo jako návod pro spolužáky z nižších ročníků.

Žáci:
Zapisují si zadání:

- ✓ *vytvořit složky pro každý předmět v průzkumníkovi*
- ✓ *hledat stránky s tématem online dokumenty*
- ✓ *vytvořit návod postupu, využít „print screen“*
- ✓ *při přebrání textu, obrázku uvést do použitých zdrojů*
- ✓ *nalézt na internetu stránky zabývající se citací a její tvorbou*

*Vytvoří dokument „NávodOnlineDokumenty.***“*

Rozpracování

První fáze tvorby. Cílem je vytvořit první verzi, která bude sloužit pro ověření získaných informací. Získání jistoty o časovém harmonogramu, možnost úprav za nízkou cenu nákladů. Kritická fáze, od ní se bude celý tvůrčí projekt odvíjet. Založení souboru pro dokumentaci.

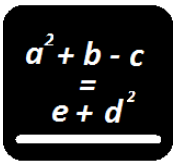
- ***Pedagog***

1. vedení k vytvoření dobrých návyků, získání sebedůvěry.
2. kontrola prvních výtvorů.
3. analýza možností a poslední úpravy plánu práce.

- ***Žáci***

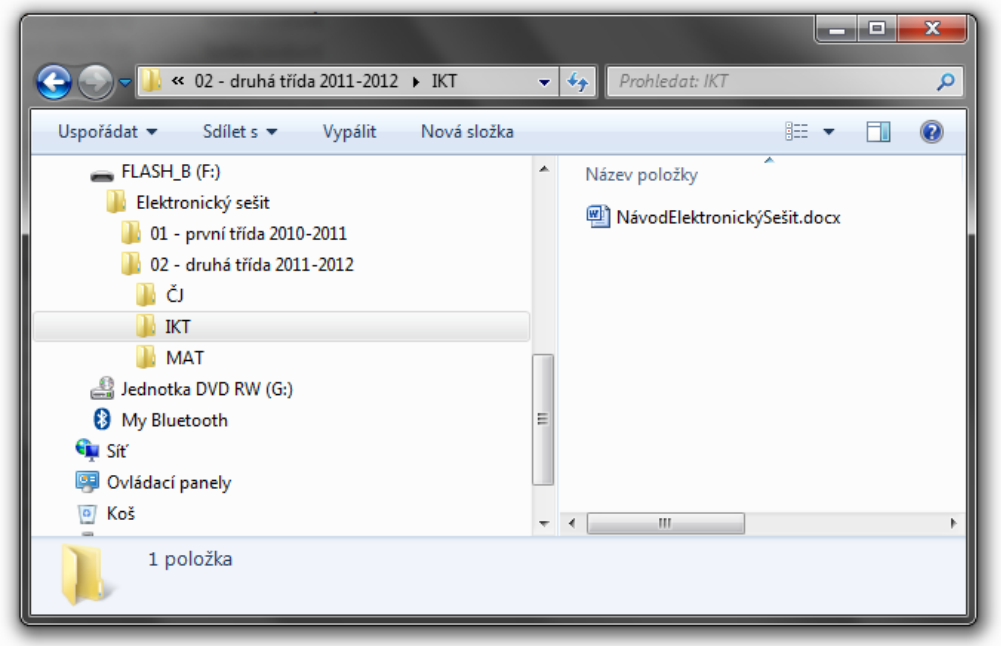
1. vytvoření pokusných instancí.
2. tvorba minimalizovaného, ale funkčního zkušebního projektu.
3. návrhy změn do plánu práce.

- ***Příklad***



Pedagog:
Kontrola vytvořených složek, celkové hierarchie.

Žáci:
Vytvoření složek v aplikaci průzkumník.
Hierarchické uspořádání:



Návrhy: pojmenování ročníků a jejich čísla, jiné uspořádání složek, …
Nalezení informací o citacích.
Nalezení serverů pro sdílení dokumentů.

Konstrukce

Cílem je splnění všech požadavků z fáze analýzy a využití informací a zkušeností z fáze návrhu. Tvorba finálního projektu. Kladen důraz na pracovní postup a funkčnost. Kontrola úplnosti dle zadání. Zkoušení, zda je projekt funkční v provozu i běžné realitě. Práce na dokumentaci.

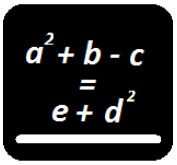
- ***Pedagog***

1. kontrola tvorby a dokumentace, zda je na ní pracováno.
2. kontrola celkového stavu projektu.
3. Zjištění, jak proběhli kritické místa tvorby.

- ***Žáci***

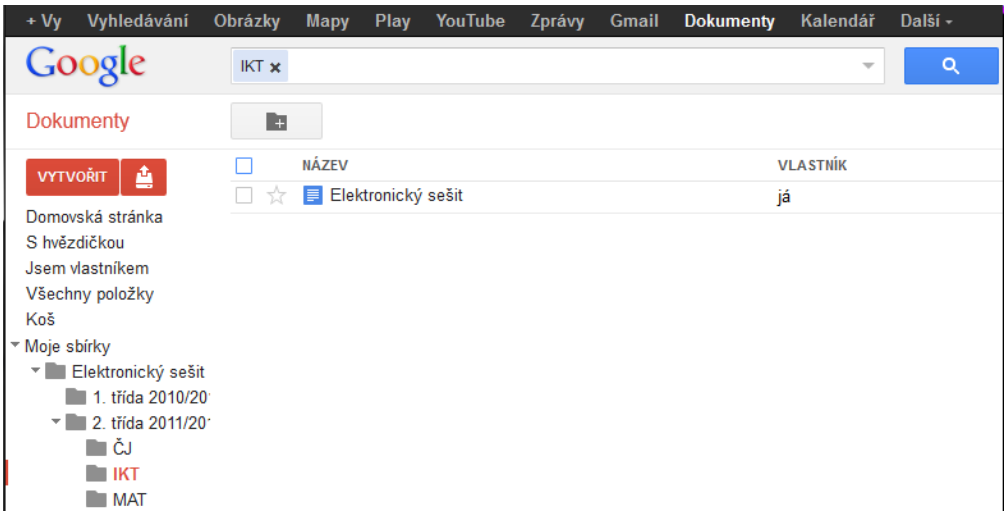
1. práce na projektu – úprava, vytvoření nápovědy.
2. doplňování dokumentace.
3. zkoušení funkčnosti a správnosti

- ***Příklad***



Pedagog:
Kontrola stavu a práce na projektu.
Konzultace problému a poradenství.

Žáci:
Registrace, nebo přihlášení do online aplikace.
Vytvoření online struktury:



Průběžné nahrávání souboru, tak aby odpovídal aktuální verzi a stavu (nebo úprava online).

Doplnění dokumentace o postup tvorby.

Kontrola funkčnosti a úplnosti.

Ukončení

Opravení všech nalezených chyb, úprava a dokončení dokumentace. Koncová revize, vytvoření krátké prezentace a hodnocení.

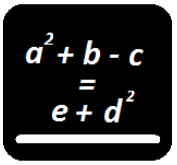
- ***Pedagog***

1. hodnocení dokumentace, softwaru a prezentace.
2. obecné hodnocení chyb a jejich náprava.
3. zopakování klíčových částí a sdělení nejvhodnějšího řešení

- ***Žáci***

1. ukončení dokumentace a vytvoření prezentace.
2. hodnocení vlastního vytvořeného projektu.
3. příprava projektu pro běžný provoz.
4. seznámení se s výtvory jiných žáků, skupin.

- ***Příklad***



Pedagog:
Hodnocení zvládnutí dokumentace.
Hodnocení vytvořeného elektronického dokumentu.
Komentář k nejlepší práci.
Komentář k často dělaným chybám.
Shrnutí jaké učivo bylo procvičeno, výhody pro použití v praxi.

Žáci:
Nahrávání do aplikace aktuální dokumentaci.
Sdílení mezi spolužáky.
Předvedení práce.
Analýza výtvorů spolužáků.
Závěrečné bilancování.

Použité zdroje:

Literatura:

- ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací*: *objektově orientovaná analýza a návrh prakticky*. Vyd. 1. Překlad Bogdan Kiszka. Brno: Computer Press, 2007, 567 s. ISBN 978-802-5115-039.

- MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003, 219 s. ISBN 80-731-5039-5.

Odkazy:

- *Rámcový vzdělávací program pro základní vzdělávání.* [online]. Praha: Výzkumný ústav pedagogický v Praze, 2007. 126 s. [cit. 2012-03-12]. Dostupné z WWW:<http://www.vuppraha.cz/wp-content/uploads/2009/12/RVPZV_2007-07.pdf>.

Vytvořeno jako příloha k diplomové práci „Tvorba softwarového projektu - metodická pomůcka pro základní a střední školy.“, v roce 2012.

Bc. Marek Zmeškal

PŘÍLOHA P III: NÁVRH METODIKY PRO SŠ

Metodická pomůcka pro tvorbu softwarových aplikací na středních školách.

Formou projektové výuky s prvky unifikovaného procesu vývoje softwaru.

Úvod

Metodická příručka je určena pedagogům středních škol. Její využití je především ve výuce informatiky, při tvorbě softwarových aplikací. Její stěžejní část je určena pro projektovou výuku tak, aby se co nejvíce přiblížila praxi. Proto je nutné mimo praktickou aplikaci obeznámit žáky s teorií a používaným softwarem.

Metodická příručka vychází z praktik projektové výuky, kterou doplňuje metodika tvorby softwaru. Výchozím modelem je unifikovaný proces. Ten je primárně koncipován pro objektové programování, jeho použitelnost je v hodná i pro projekty, které nevyužívají objektové programování. Objekty lze nahradit funkcemi a procedurami.

Výsledné požadavky jsou určeny pro gymnázia, z čehož i byly čerpány (RVP pro gymnázia). Využití metodiky může být i na jiných školách, na kterých se vyučuje programování a žáci mají možnost programovat jak jednotlivě, tak i ve skupinách.

Cílem této metodiky je spojení výuky s praxí. Hlavní myšlenkou je využití UP (unifikovaný proces), který je založen na rigorózním přístupu s iterativní implementací a tím se i nejvíce se přibližuje běžné výuce.

Usnadnění pochopení algoritmického myšlení je spojeno s pohledy UML(Unifikovaný Modelovací Jazyk). Ten nabízí různé náhledy na řešenou situaci, problém. Další využití UML pohledů je v dokumentaci a následně i prezentaci. Metodika není stavěna pro konkrétní programovací jazyk ani pro konkrétní pomocné aplikace.

Obsah

Výstupní požadavky dle RVP

Vybrané požadavky, které jsou pro danou metodiku důležité.

Žák:

- aplikuje algoritmický přístup k řešení problémů
- ovládá, propojuje a aplikuje dostupné prostředky ICT
- zpracovává a prezentuje výsledky své práce s využitím pokročilých funkcí aplikačního softwaru, multimediálních technologií a internetu
- využívá teoretické i praktické poznatky o funkcích jednotlivých složek hardwaru a softwaru k tvůrčímu a efektivnímu řešení úloh
- využívá dostupné služby informačních sítí k vyhledávání informací, ke komunikaci, k vlastnímu vzdělávání a týmové spolupráci
- posuzuje tvůrčím způsobem aktuálnost, relevanci a věrohodnost informačních zdrojů a informací
- využívá nabídku informačních a vzdělávacích portálů, encyklopedií, knihoven, databází a výukových programů
- využívá informační a komunikační služby v souladu s etickými, bezpečnostními a legislativními požadavky

Tvorba softwarového produktu

Dle metodiky unifikovaného procesu je rozdělena i tato příručka. Fáze na sebe vzájemně a plynule navazují, přičemž u každé je jasně daný cíl. Dále jsou uvedeny úkoly, které by měly být plněny především žáky i pedagogy. Jednotlivé části jsou koncipovány tak, aby zpřehlednily a ujasnily práci na daném projektu a přinesly především žákům přehled a základní informace o postupu práce při tvorbě softwarového produktu. Před započetím je nutné, aby žáci měli dostatečné teoretické znalosti nebo bylo možné tyto znalosti získat během práce na jejich díle.

Role jsou dány přirozenou pozicí, která je obvyklá pro běžnou výuku v České republice. Procenta označují celkovou dobu z celého projektu. Minimální doporučenou dobou na projektu jsou čtyři vyučovací hodiny, pokud má být projekt dostatečně zpracován.

Každá fáze se skládá z několika částí (požadavky, analýza, návrh, implementace, testování, dokumentace), které postupným opakováním dosáhnou přírůstku aplikace. Každá fáze je ukončena splněním dílčích úkolů.

Zahájení

Čas - 15-20%

Je to počáteční fáze, jedná se o odstartování celého projektu. Seznámení s požadavky a tématem. Motivace pro konečné použití softwaru a shrnutí nových poznatků, které budou prohloubeny a vyzkoušeny.

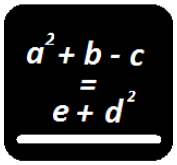
- ***Pedagog***

1. Sdělí požadavky na projekt, funkční požadavky, nefunkční požadavky, podmínky.
2. Informuje o tom, kdo, kdy a jak bude projekt využívat.
3. Rozpracuje hlavní slovní označení pro slovník projektu.
4. Návrh pracovního rozvrhu.
5. Rozebrání možných rizik.
6. Náčrt architektury softwaru.

- ***Žáci***

1. Navrhují, upravují požadavky na projekt.
2. Zpracování počátečních případů užití do formy matice.
3. Vytvoření slovníku projektu – jména funkcí, procedur, proměnných.
4. Schválení pracovního rozvrhu.
5. Nalezení a konzultace pracovních rizik.
6. Tvorba počátečních dokumentů pro dokumentaci.

- ***Příklad***



Pedagog:

Zadání:

Tvorba základní aplikace, kde na úvodní obrazovce bude menu, které bude možné procházet šipkami nahoru a dolů. Volba bude zvýrazněna a spouštět se bude klávesou „Enter“. Bude možné dále do tohoto menu přidávat v průběhu celého roku další vytvořené programy. Po přidání bude možné programy spouštět. Tento softwarový projekt je určen jak pro žáky, pro přehled všech vytvořených programů jako portfolio, tak i pro rodiče, kteří budou mít možnost při třídních schůzkách spustit a podívat se na vaši tvorbu. Čas vyhrazený na celý projekt je čtyři vyučovací hodiny. Z toho poslední hodina bude prezentační. Výsledkem je spustitelný a odladěný projekt s dokumentací. Možná rizika jsou v souvislosti s načítáním kláves, popřípadě po vložení dalších programů možná nefunkčnost.

Žáci:

Vytvoření v kancelářské aplikaci (Word, Poznámkový blok, GoogleDOCS, …) soubor, ve kterém bude dokumentace k projektu. Obsahuje veškeré náležitosti, jako je:

- ✓ *identifikace žáka (jméno, příjmení, třída, rok tvorby, …)*
- ✓ *název projektu*
- ✓ *zadání projektu*
- ✓ *plán projektu*
- ✓ *různé pohledy na danou problematiku*
- ✓ *závěr – shrnutí problémových částí a jejich řešení*

Vytvoření matice požadavků a funkcí pokrývající dané požadavky.

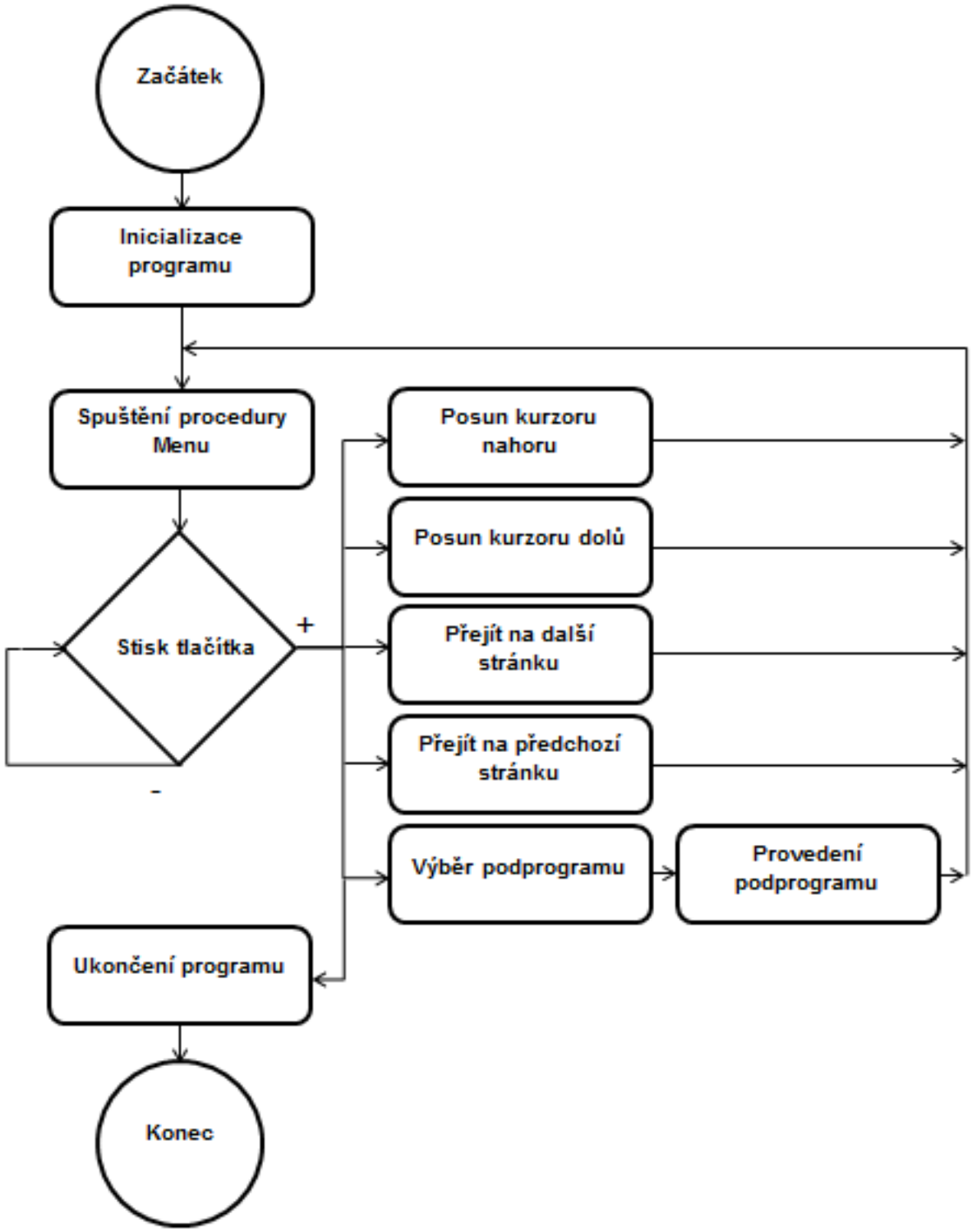
Matice požadavků a procedur	*procedura Up*	*procedura Down*	*procedura Spust*	*procedura Menu*
posun nahoru	X			
posun dolů		X		
výběr programu			X	
celkové ukončení				X
nápověda				X

Slovník projektu:

Název	**Parametry**	**Úkol**
PortfolioJmenoPrijmeni	-	Jméno programu.
Menu	-	Hlavní procedura programu.
pozice	Integer	Proměnná procedury menu, určuje, na které pozici se nachází kurzor
Up	pozice:Integer	Posune a zvýrazní kurzor nad předchozí pozicí, pokud je na počátku skočí na konec.
…	**…**	**…**

Nalezení návrhového vzoru (pokud žáci mají vytvořenou vlastní databázi v rámci třídy nebo školy).

Obecný diagram průběhu programu:



Rozpracování

Čas - 20-30%

První fáze tvorby spustitelného kódu. Cílem je první spustitelný systém, jedná se o první pokus o funkční program. Jedná se o kritickou fázi, od ní se bude celý systém odvíjet.
Iterativním opakováním částí a vytváření tak přírůstku je typické pro fázi rozpracování:

Požadavky	- upřesnění rozsahu a požadavků
Analýza	- stanovení objemu celkové tvorby
Návrh	- vytvoření stabilní architektury
Implementace	- tvorba spustitelného základu
Testování	- testování základu
Dokumentace	- vytvoření osnovy a životního cyklu

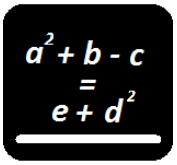
- ***Pedagog***

1. Vedení k vytvoření dobrého, robustního základu.
2. Kontrola modelů – statický, dynamický.
3. Analýza možností a poslední úpravy plánu práce.

- ***Žáci***

1. Vytvoření statického a dynamického modelu.
2. Tvorba spustitelného základu.
3. Návrhy změn do plánu práce.

- ***Příklad***



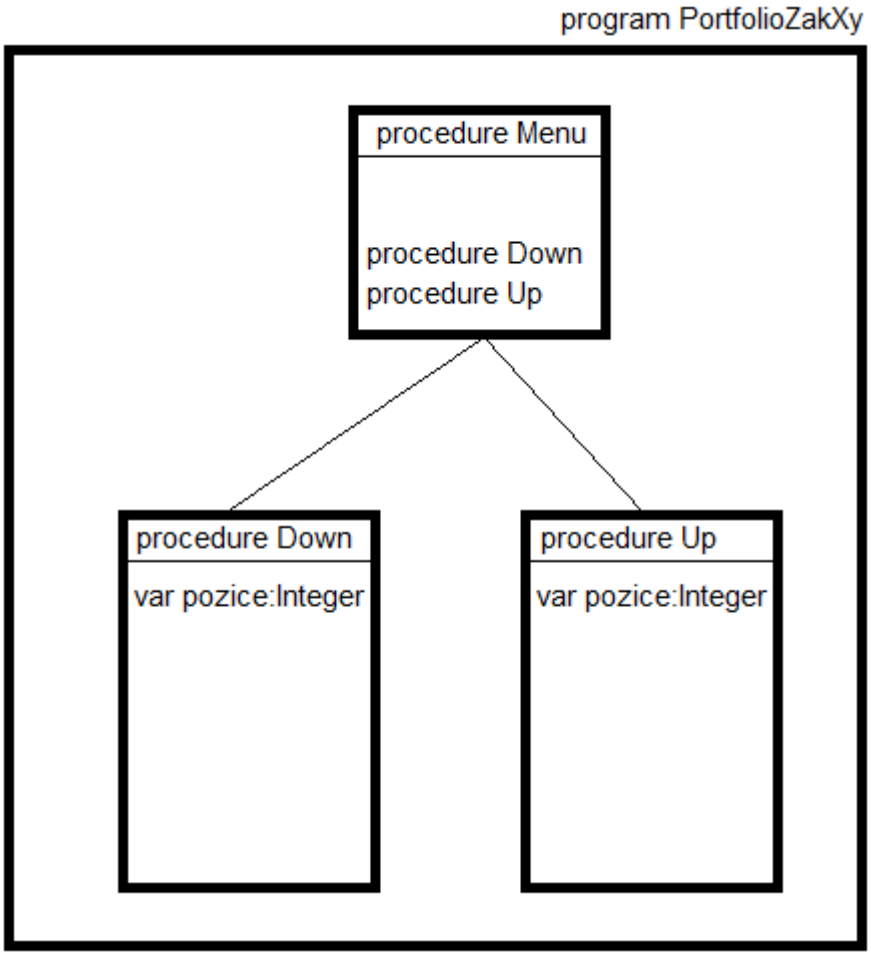
Pedagog:

Kontrola tvorby, brždění přehnaných invencí, omezení nadměrné byrokracie, kontrola kompaktnosti modelů a dosud vytvořených pohledů.

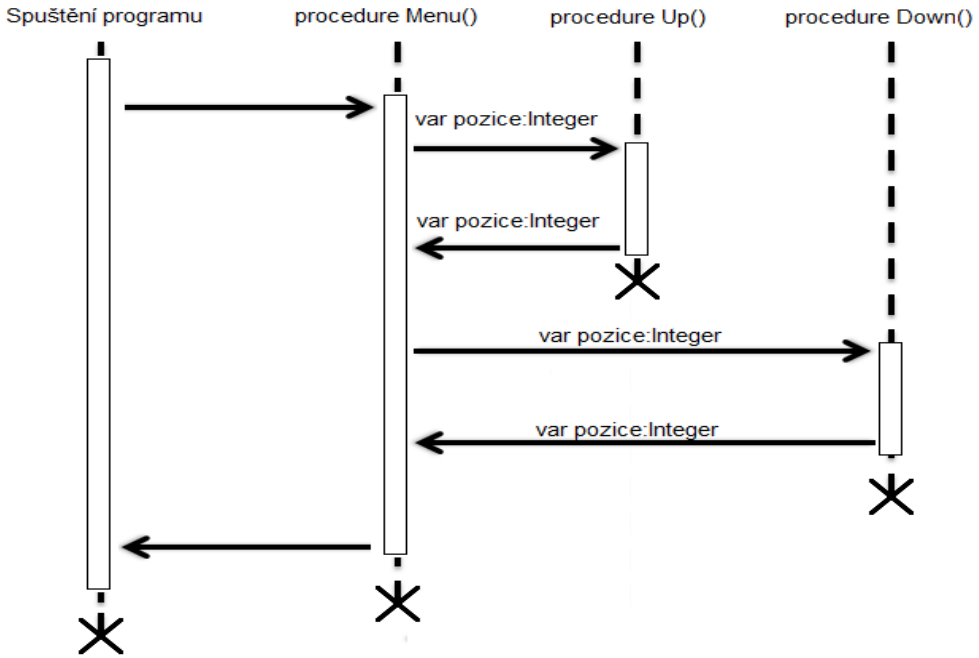
Žáci:

Vytvoření pohledů na řešený problém (využití kancelářského balíčku – malování, Word, Excel, nebo i jiné programy k tomu určené - Enterprise Architect).

Statický model:



Dynamický model:



Implementace kódu:

- ✓ *kostra zdrojového kódu (vytvoření funkcí a přidání parametrů)*
- ✓ *komentáře ke zdrojovému kódu (komentář ke každé proceduře, funkci, parametru, globální proměnné, lokální proměnné, datovým strukturám – jaký je úkol, typ, očekávané výstupy)*

Ukázka v prostředí Pascal (komentáře ve složených závorkách „{“ „}“), pokud používáme prostředí jiné, například Visual Studio pro C#, využíváme vestavěné příkazy a linky na jednotlivé proměnné… (///<summary>, <param name=“ “>, <returns>…) jenž umožňují přímí export do dokumentační podoby.

```
program PortfolioZakXy;
{*** Název programu *PortfoliZakXy*}
{Vytvoření halvního menu, ve keterém
 je možné spouštět a přidávat další
 programy během celého školního roku}

{*** Procedura *Down*}
{Nastavuje kurzor na další pozici směrem dolů,
 při dosažení poslední části skočí opět na
 první pozici na hoře.}
{Parametry funkce Down -
  pozice:Integer - aktualizuje a uchovává aktuální pozici kurzoru}
procedure down(var pozice:Integer);
begin

end;
{*** Procedura *Menu*}
{ Hlavní a první procedura, která zobrazuje menu,
stará se o výběr a spuštění programu z nabídky.}
{Parametry funkce Menu - bez parametrů}
procedure Menu();
var pozice:Integer; {lokální proměnná *pozice* typu *Integer* | ukládá pozici kurzoru do doby
                                                              než je spuštěn vybraný program}
begin

end;
{*** Hlavní program ***}
begin
  Menu();
end.
```

Konstrukce

Čas - 30-50%

Cílem je splnění všech požadavků z fáze analýzy a návrhu. Vytvoření konečné verze systému. Kladen důraz na pracovní postup implementace.

Pro fázi konstrukce je typické provádět iteračně části s následným zaměřením:

Požadavky	- odhalení přehlédnutých požadavků.
Analýza	- dokončení statického modelu
Návrh	- dokončení dynamického modelu
Implementace	- zajištění provozní způsobilosti
Testování	- testování tvořených variant
Dokumentace	- úpravy a opravy vytvořené dokumentace dle aktuálního stavu

- ***Pedagog***

1. Kontrola modelů a dokumentace dle aktuálního stavu.
2. Kontrola celkového stavu projektu.

- ***Žáci***

1. Dopracování modelů dle aktuálního stavu aplikace
2. Práce na projektu – zdrojový kód, úprava prostředí, vytvoření nápovědy.
3. Vytvoření uživatelské příručky.
4. Testování aplikace.

- ***Příklad***

Pedagog:
Kontrola plnění plánů, stavu a dokumentace, zda je ve shodě s realitou.
Žáci:
Implementace zdrojového kódu.
Dodělání restů z předchozí fáze (přehlédnuté věci, nedostatky).
Aktualizace dokumentů, pohledů dle tvorby.
Vzájemné testování aplikace.
Oprava chyb.

Ukončení

Čas - 15-20%

Opravení všech nalezených chyb, úprava a dokončení dokumentace a manuálů. Koncová revize, vytvoření krátké prezentace a hodnocení.

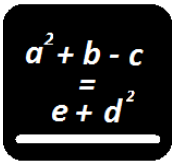
- ***Pedagog***

1. Hodnocení dokumentace, softwaru a prezentace.
2. Obecné hodnocení chyb a jejich náprava.

- ***Žáci***

1. Ukončení dokumentace a vytvoření prezentace.
2. Hodnocení projektu.
3. Příprava aplikace pro běžný provoz.
4. Seznámení se výtvory jiných žáků, skupin.

- ***Příklad***



Pedagog:

Vlastní hodnocení jednotlivých projektů.
Popis nejlepší práce, jak by to mělo být nejlépe udělané.
Ukázka chyb, které žáci provedli a jejich možné odstranění.
Řízení prezentací a kontrola časových limitů.

Žáci:

Vytvoření prezentace na maximálně pět minut – stručně vyzvednou to, co se jim povedlo, jaké nastaly během tvorby potíže, jak je vyřešili a proč, návrhy na lepší řešení.
Hodnocené vlastní práce.
Hodnocení práce ostatních žáků.
Můžou vytvořit vlastní návrhový vzor, který by mohli dále využívat. (Publikovat v rámci třídy na sdílených dokumentech apod.)

Použité zdroje:

Literatura:

- ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací*: *objektově orientovaná analýza a návrh prakticky*. Vyd. 1. Překlad Bogdan Kiszka. Brno: Computer Press, 2007, 567 s. ISBN 978-802-5115-039.
- GAMMA Erich, HELM Richard, JOHNSON Ralph, VLISSIDES John. *Návrh programů pomocí vzorů*: *Stavební kameny objektově orientovaných programů*. Praha: Grada Publishing, 2003. ISBN 80-247-0302-5.
- MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003, 219 s. ISBN 80-731-5039-5.

Odkazy:

- *Rámcový vzdělávací program pro gymnázia.* [online]. Praha: Výzkumný ústav pedagogický v Praze, 2007. 100 s. [cit. 2012-03-12]. Dostupné z WWW: <http://www.vuppraha.cz/wp-content/uploads/2009/12/RVPG-2007-07_final.pdf>. ISBN 978-80-87000-11-3.
- KRAVAL, Ilja. *Objektové modelování pomocí UML v praxi, 2005* [online]. ©2005 [cit. 2012-03-17]. Dostupné z: http://www.objects.cz/

Vytvořeno jako příloha k diplomové práci „Tvorba softwarového projektu - metodická pomůcka pro základní a střední školy.“, v roce 2012.

Bc. Marek Zmeškal