Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Metódy a nástroje pre výučbu softvérového inžinierstva v predmetoch o programovaní

Diplomová práca

2014 **Bc. Peter Pajkoš**

Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Metódy a nástroje pre výučbu softvérového inžinierstva v predmetoch o programovaní

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1 Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	doc. Ing. Jaroslav Porubän, PhD.
Konzultant:	doc. Ing. Jaroslav Porubän, PhD.

Košice 2014 **Bc. Peter Pajkoš**

Abstrakt v SJ

Hľadanie nových foriem výučby predmetov o programovaní tak, aby v sebe prirodzene obsahovali aj procesy, ktoré prebiehajú pri vytváraní softvéru je náplňou tejto diplomovej práce. Práca sa zaoberá využívaním moderných nástrojov projektovej komunikácie v procese výučby a integráciou týchto nástrojov s už existujúcimi aplikáciami pre podporu výučby predmetov o programovaní. V práci sú charakterizované základné prístupy k výučbe programovania a možnostiach výučby procesov prebiehajúcich v softvérovom inžinierstve v takých kurzoch. Tieto prístupy sú kľúčové pre pochopenie štruktúry kurzov na univerzitách definujúc obmedzenia, v ktorých vznikala aj táto práca. Pokusy o zlepšenie aktuálnej situácie v oblasti výučby programovania, ich prínosy aj záporné stránky sú popísané v analytickej časti práce. Výstupom teoretickej časti je návrh riešenia integrácie procesov softvérového inžinierstva s výučbou programovania v konkrétnom kurze, inšpirovaný prácami kolegov a vychádzajúci z aktuálneho stavu výučby kurzu. V časti venovanej implementácii sú opísané prístupy k prepojeniu programovania projektu s projektovou komunikáciou, možnosti kontroly postupu študenta a reakcia softvéru na tento postup. Použiteľnosť predkladaného riešenia je zhodnotená na množine študentov, ktorých postrehy a pripomienky sú podkladom pre zhodnotenie prínosov práce.

Kľúčové slová

výučba programovania, procesy, softvérové inžinierstvo

Abstrakt v AJ

A new approach towards university programming courses education, combining processes commonly found in software engineering with programming learning is the main issue of the submitted diploma thesis. The thesis focuses on using modern, up-to-date project communication tools in programming courses education and integrating those tools with already existing applications for supporting the education. Widely accepted approaches towards programming education and possible options for teaching software engineering processes within programming courses are described in the paper, as well. The mentioned approaches are crucial for understanding the structure of university programming courses and its limits, present during developing of the thesis. The analytic part of the work also deals with attempts to improve current situation in programming education, their assets and negatives alike. The theoretical part of the thesis provides us with a possible solution for integrating software engineering processes with programming education within a specific course. The solution itself is inspired by the approaches of colleagues and current state of education present in the mentioned course. The part dedicated to implementation describes connecting project development to project communication tools, possibilities for controlling student‘s progress and responding to that progress. The proposed solution was tested in association with several students whose feedback proved extremely valuable when evaluating the assets of this thesis.

Kľúčové slová v AJ

programming education, processes, software engineering

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **9.2.1 Informatika**
Študijný program: **Informatika**

Názov práce:

Metódy a nástroje pre výučbu softvérového inžinierstva v predmetoch o programovaní
Methods and Tools for Software Engineering Education in Programming Courses

Študent:	**Bc. Peter Pajkoš**
Školiteľ:	**doc. Ing. Jaroslav Porubän, PhD.**
Školiace pracovisko:	**Katedra počítačov a informatiky**
Konzultant práce:	
Pracovisko konzultanta:	

Pokyny na vypracovanie diplomovej práce:

1. Analyzovať existujúce prístupy k výučbe softvérového inžinierstva v predmetoch o programovaní.
2. Navrhnúť a implementovať model virtuálnej spoločnosti pre simuláciu procesov prebiehajúcich pri vývoji softvéru.
3. Integrovať navrhnutý model s používanými nástrojmi pre podporu výučby predmetov o programovaní.
4. Experimentálne overiť použiteľnosť vytvoreného riešenia a vyhodnotiť prínosy v rámci vyučovania.
5. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 02.05.2014
Dátum zadania diplomovej práce: 31.10.2013

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -11-

doc. Ing. Jaroslav Porubän, PhD.
vedúci garantujúceho pracoviska

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som diplomovú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice 2. 5. 2014 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste chcem vyjadriť úprimnú vďaku všetkým ľuďom, ktorí sa na vytváraní tejto práce priamo i nepriamo podieľali. Ďakujem pánovi Alexandrovi Loziakovi za jeho nesmiernu pomoc pri formovaní charakterov virtuálnych osôb vytvorených v diplomovej práci, bez pomoci ktorého by dané osoby boli len bezduchými premennými programu. Veľká vďaka patrí piatim odvážnym študentom, ktorí sa podieľali na testovaní navrhovaného riešenia a poskytli mi cennú spätnú väzbu pre ďalšie vylepšovanie mojej práce. Ďakujem teda slečnám Lenke Dzurjakovej a Márii Véghovej aj pánom Petrovi Dobozovi, Michalovi Jurečkovi a Mariánovi Harčaríkovi.

Rodičom, bratom, priateľom a blízkym ďakujem za pomoc, cenné rady a podporu ako pri písaní tejto práce, tak aj počas celého štúdia na vysokej škole, ktorého vyvrcholením je práve táto práca.

Pánovi Ing. Miroslavovi Biňasovi, Phd. patrí moja vďačnosť za prístup na školskú verziu systému GitLab a za pripomenutie, že s veľkou mocou prichádza veľká zodpovednosť. Zodpovednosť, ktorú mi do rúk zveril výnimočný človek i pedagóg. Človek, ktorý verí, že učiť iných je poslanie i radosť a dokazoval to počas celej tvorby tejto práce. Nesmierna vďaka patrí doc. Ing. Jaroslavovi Porubänovi, PhD.

Predhovor

Predkladaná záverečná práca pojednáva o nových postojoch k výučbe predmetov o programovaní na vysokých školách. Kľúčovou myšlienkou práce je vniesť procesy, ktoré sa objavujú pri vytváraní softvéru do vyučovania programovania, aby tak študenti boli schopní pochopiť miesto implementácie aplikácie v kontexte celého vývoja softvéru.

Chýbajúce vedomosti z oblasti projektovej komunikácie, neuvážené programovanie bez pochopenia hlbších súvislostí a väzieb v aplikácii patria medzi najčastejšie nedostatky čerstvých absolventov vysokých škôl v oblasti informatiky, na čo poukazujú aj rôzne spoločnosti pôsobiace v tomto odbore. Spomínané nedostatky sú hlavnou motiváciou tejto práce, ktorá prináša konkrétne riešenie tohto aktuálneho problému.

Poskytnuté riešenie je z hľadiska výskumného charakteru práce zamerané na konkrétny predmet štúdia, pričom využíva už úspešne implementované nástroje pre podporu výučby a snaží sa tieto nástroje zlúčiť do harmonického celku. Nenúteným a prirodzeným spôsobom dochádza k zmene formy výučby tak, aby zahŕňala procesnú stránku vývoja aplikácií, z čoho benefitujú študenti i lektori.

Obsah

1 Úvod **1**

1.1 Ciele diplomovej práce 2

2 Softvérové inžinierstvo v univerzitnom prostredí **4**

3 Univerzitné prostredie stimulujúce študentov **9**

3.1 Využívanie softvérových nástrojov vo výučbe SI 11

3.2 Zavedenie SCRUM do výučby SI 13

3.3 Ďalšie aktivity smerujúce k vyššej kvalite výučby SI 15

4 Inovatívna výučba predmetov o programovaní **19**

4.1 Potreba externého manažéra v procese výučby 23

4.2 Potreba vytvorenia rozsiahleho tímu v procese výučby 24

5 Odpoveď psychológie na modelovanie osobnosti **25**

5.1 Využitie modelu OCEAN vo výučbe predmetov o programovaní . . . 28

5.2 Vytvorenie profilov členov virtuálnej firmy pre podporu výučby 32

6 Od nápadu k produktu **35**

6.1 Pozícia študenta v projekte integrovania procesov SI do výučby predmetov o programovaní 36

6.2 Metódy a postupy získania študentských riešení 37

6.3 Metódy a postupy vyhodnotenia študentských riešení 40

6.4 Metódy a postupy vypracovania stratégie ďalšieho postupu vývoja softvéru 41

6.5 Sociálna interakcia v systéme riadenia postupu študenta 43

7 Od produktu k nápadom **46**

7.1 Študeni v úlohe účastníkov projektu automatizovanej kontroly postupu študentských riešení 48

7.2 Študeni v úlohe bádateľov nových prístupov k zlepšeniu výučby . . . 52

8 Záver **54**

Zoznam použitej literatúry **55**

Zoznam príloh **59**

Zoznam obrázkov

2 – 1 Oblasť záujmu informatiky. Prevzaté z (ACM & IEEE, 2013). 6
2 – 2 Oblasť záujmu softvérového inžinierstva.Prevzaté z (ACM & IEEE, 2013). 7
3 – 1 Priebeh štúdia kurzu SI v Nemecku. Prevzaté z (Scharf, Koch, 2013). 13
3 – 2 štúdiovo-orientovaný model. Prevzaté z (Sureka, Gupta, Sarkar, Chaudhary, 2007). 17
4 – 1 Konceptuálny návrh nástrojovej podpory pre výučbu procesov softvérového inžinierstva 22
6 – 1 Ukážka obsahu súboru builds.xml 41
6 – 2 Ukážka úlohy v systéme GitLab 42
6 – 3 Ukážka pochvalnej korešpondencie 45
6 – 4 Ukážka korešpondencie pri zlyhávaní študenta 45
7 – 1 Ukážka prostredia, v ktorom pracuje študent 46

Zoznam tabuliek

5 – 1 Osobnostné profily zamestnancov virtuálnej firmy 32
7 – 1 Odpovede študentov z dotazníka o spokojnosti s testovaným systémom 50
7 – 2 Legenda k odpovediam študentov 51

Zoznam symbolov a skratiek

I	Informatika
IDE	Integrované vývojové prostredie
IS	Informačné systémy
IT	Informačné technológie
PI	Počítačové inžinierstvo
SI	Softvérové inžinierstvo

Slovník termínov

Informatika vedná disciplína zaoberajúca sa zberom, triedením, spracovaním, vyhodnocovaním, využívaním, uchovávaním a odosielaním informácií.

Proces je postupnosť či rad časovo usporiadaných udalostí tak, že každá predchádzajúca udalosť sa zúčastňuje na determinácii nasledujúcej udalosti.

Programovanie činnosť, ktorá zahŕňa procesy vytvárania algoritmov a ich zápis v programovacom jazyku do programov, testovanie a ladenie vytvorených programov ako aj vypracovanie dokumentácie.

Softvérové inžinierstvo predstavuje využitie systematického, riadeného a merateľného prístupu k návrhu, tvorbe a spravovaniu softvérových produktov.

1 Úvod

„Ak máte univerzitný diplom, môžete si byť absolútne
istý jednou vecou... máte univerzitný diplom."
Neznámy autor

Pripravenosť študentov informačných technológií pre prax je v súčasnosti často krát predmetom vášnivých diskusií, nie len v rámci firiem a skúsených členov projektových tímov, ktorí akoby zabúdali na svoje začiatky, ale aj v rámci celej spoločnosti. Buďme ale objektívni a uznajme, že nie všetkým zamestnancom IT firiem musíme pripomínať ich kariérne začiatky, keďže niektorí jedinci sa stali kvalitnými programátormi už počas štúdií, vynaložením vlastného úsilia a podporou či už učiteľov alebo svojpomocne. Pokračujúc nastoleným objektívnym pohľadom však uznajme, že máme aj vynikajúcich pedagógov za katedrami v našich aulách. Vynára sa teda otázka ako tieto diskusie zmierniť, ako vychovať kvalitných programátorov, ktorí budú zároveň výbornými softvérovými inžiniermi, a teda ich diplom bude mať skutočnú výpovednú hodnotu o ich kvalitách.

Aktuálny prístup k výučbe študentov zaoberajúcich sa informatickými študijnými programami je mnohokrát taký, že výučba programovania je oddelená od procesnej stránky tvorby softvéru. Práve túto skutočnosť je možné vyčítať vysokým školám, pretože pri vývoji softvérových systémov v praxi sú princípy, technológie a procesy silne previazané a systematicky aplikované počas celého trvania vývoja softvéru. Uvedomenie si týchto nedostatkov vo vyučovacom procese je kľúčové pre zvýšenie úrovne vedomostí študentov informačných technológií.

1.1 Ciele diplomovej práce

Berúc na vedomie aktuálny stav problematiky výučby programovania a softvérového inžinierstva, práca sa zameriava na hľadanie inovatívnej metódy a formy vyučovania predmetov o tvorbe softvéru, ktorá zabezpečí prepojenie vyučovania programovania, technológií a procesov vývoja softvéru do harmonického celku. Pozornosť bude zameraná na predmet druhého ročníka bakalárskeho štúdia, konkrétne predmet Technológie JAVA. Pri zachovaní obsahu cvičení, ktorých náplňou je úspešné implementovanie známej hry Míny sa záujem obráti na inováciu formy organizácie cvičení. Od plnenia zoznamu úloh zobrazených na webovej stránke sa presunieme k simulácií procesov softvérového inžinierstva počas vytvárania softvérového produktu, teda hry Míny. Tak ako vo firme zaoberajúcej sa vývojom softvéru, tak aj na cvičeniach zvolíme prístup ku študentovi ako ku plnohodnotnému členovi projektového tímu, v ktorom samozrejme nemôže chýbať projektový manažér zodpovedný za dozor nad plnením úloh a postupom v rámci zadefinovaného procesu. Pre dosiahnutie ešte väčšieho priblíženia sa k realite, tím ľudí bude pozostávať z väčšieho počtu osôb. Problémom ostáva nájdenie skutočnej firmy, ktorá by bola ochotná spolupracovať so školou vo vyučovacom procese. Riešenie sa naskytá v podobe projektového tímu, v ktorom všetci členovia, s výnimkou študenta, sú čírou simuláciou a fikciou, čomu bude podriadená navrhovaná nástrojová podpora.

V priebehu vypracovávania projektu bude študent vedený svojim tímom aj lektorom, aby na konci semestra projekt úspešne implementoval. S tým je nevyhnutne spätých niekoľko záležitostí, akými sú automatické spracovanie štrukturálnych a sémantických chýb v zdrojovom kóde študenta, prideľovanie nových úloh študentom, aby mohli ďalej v projekte napredovať a podobne. Navrhovaná ako aj existujúca nástrojová podpora odbremení učiteľov od netvorivých a časovo náročných úloh. Sledovaním postupu študenta bude lektor schopný efektívne rozdeliť svoj čas na hodine medzi študentov tak, aby samostatne pracujúcich študentov zbytočne nepou-

čoval o záležitostiach im známym, a naopak, aby sa venoval dostatočne tým študentom, ktorí to budú potrebovať. Lektor bude schopný pozorovať aj to, ktoré učivo je pre poslucháčov náročnejšie, čo je možné využiť pri neustálom zlepšovaní štúdijných materiálov.

Najpodstatnejší je prínos tejto diplomovej práce pre študenta, ktorý získa predstavu o dianí vo firmách operujúcich v oblasti informačných technológií od odovzdávania svojho riešenia do systému pre správu verzií, cez automatickú spätnú väzbu pri kompilácii svojho zdrojového kódu až po označenie rozpracovanej úlohy ako vyriešená v systéme úloh. Tradičná otázka od poslucháčov „ Načo mi v praxi bude tento predmet?“ sa stane irelevantnou a v prípade úspešných výsledkov tejto práce a jej aplikácií na ostatné predmety o programovaní by mohla škola vzdelávať kvalitných odborníkov pre prax už v bakalárskom stupni štúdia. Splnením týchto cieľov tak citát z úvodu stratí nádych irónie.

2 Softvérové inžinierstvo v univerzitnom prostredí

„Softvérové inžinierstvo predstavuje využitie
systematického, riadeného a merateľného
prístupu k návrhu, tvorbe a spravovaniu
softvérových produktov."
(IEEE, 1990)

Držiac krok s aktivitami spojenými s tvorbou serióznych softvérových systémov, problémy späté s vývojom týchto produktov sa neprestávajú vynárať. A vynárali sa už v samotných začiatkoch tvorby programov, dôkazom čoho je konferencia NATO z roku 1968, organizovaná najmä kvôli tejto téme a s výsledkom v podobe zavedenia návrhu softvérového inžinierstva (Naur & Randell, 1969). Uvedená konferencia stojí za zmienku predovšetkým kvôli identifikovaniu veľkého rozsahu tém prítomných ako kľúčové v modernom softvérovom inžinierstve. Rozmer a komplexnosť systémov, problémy v odhadovaní nákladov rôzneho charakteru a predpokladaní aktivít, či aj možnosť vytvárania systémov na báze komponentov boli konzultované pri tejto príležitosti. (Hislop, 2009) pripomína, že vybrané otázky boli dobre pochopené prinajmenšom v prístupe využívajúcom vodopádový model.

Naproti všeobecnému akceptovaniu výsledkov a podpore aktivít softvérového inžinierstva, otázka absolútnej samostatnosti softvérového inžinierstva ako disciplíny nenachádza jednoznačnú odpoveď. V súčasnosti, viac ako 45 rokov od konferencie v Nemecku, však existuje väčšinou odborníkov prijatý konsenzus pre organizáciu znalostí v oblasti informatiky a systém profesií v tejto oblasti. V akademickom prostredí sa pri diskusiách o študijných odboroch najčastejšie spomína vzťah medzi softvérovým inžinierstvom a informatikou(odbor Computer science). Vzťahy medzi jednotlivými študijnými odbormi sa ilustrujú pomocou vennových diagramov, kde sa skúma, či jednotlivé disciplíny majú spoločný prienik, sú úplne nezávislé alebo jedna je podmnožinou druhej. Na základe toho sa potom rozhodne, či daná univerzita vy-

učuje softvérové inžinierstvo ako samostatný odbor.(ACM & IEEE, 2013) ponúka vo svojej práci uznávanú reprezentáciu jednotlivých štúdijných odborov v rámci informatiky, kde jasne poukazuje na odlišnosť medzi nimi, na druhej strane však ilustruje ich vzájomné prepojenie.

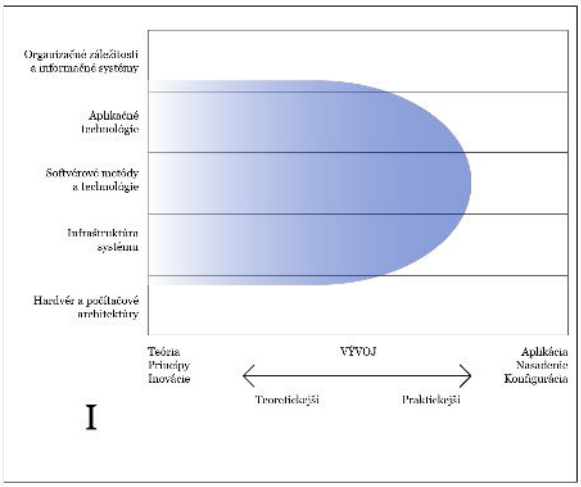
Štúdia bola zameraná na päť majoritných disciplín v rámci odborov, ktoré pre dosiahnutie svojich cieľov potrebujú výpočtovú techniku, prípadne z nej benefitujú alebo ju vytvárajú. Konkrétne sú to počítačové inžinierstvo(PI), informatika (I), informačné systémy(IS), informačné technológie(IT) a softvérové inžinierstvo(SI). Zo spomínaných dôvodov pri zavádzaní softvérového inžinierstva medzi štúdijné odbory na vysokých školách, zameriame našu pozornosť práve na kontrast medzi SI a I.

Informatika pokrýva široký rozsah od teoretických základov a algoritmizácie až po najnovší vývoj v robotike, inteligentných systémoch a podobne. Práca odborníka v tejto oblasti spočíva v nasledujúcich činnostiach:

- navrhovanie a implementácia softvéru,
- skúmanie nových možností aplikovania počítačov do iných oblastí života,
- vytváranie efektívnejších algoritmov pre riešenie problémov v oblasti výpočtovej techniky.

Študijný plán disciplíny I musí obsiahnuť uvedenú rozsiahlu oblasť (Obr. 2 – 1) a to je dôvod, prečo je odbor kritizovaný pre zlú pripravenosť absolventov pre špecifické povolania. Na druhej strane je absolvent počas štúdia oboznámený so základmi vedy v širšom kontexte, rozumie zákonitostiam a väzbám medzi jednotlivými oblasťami v rámci výpočtovej techniky a to ho predurčuje na jednoduché pochopenie neznámej oblasti a rýchlu adaptáciu.

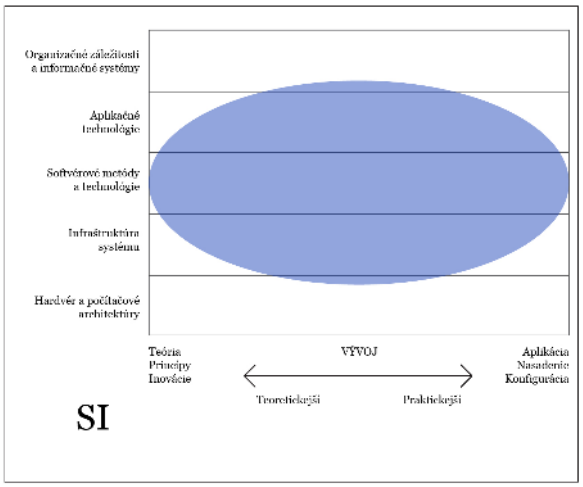
Softvérové inžinierstvo sa zaoberá vývojom a spravovaním spoľahlivých a efektívnych



Obr. 2 – 1 Oblasť záujmu informatiky. Prevzaté z (ACM & IEEE, 2013).

softvérových produktov, pričom ich vývoj a údržba je cenovo dostupná a výsledný produkt spĺňa požiadavky zadávané zákazníkom. Nehmotná povaha softvérového produktu ako aj nesúvislá povaha softvérových operácií odlišujú SI od iných inžinierskych disciplín. Po náplňovej stránke sa snaží o integrovanie matematických princípov a počítačovej vedy s inžinierskymi praktikami pre vývoj softvéru cielený na hmotné prostriedky vo forme počítačov, výrobných liniek a podobne. Záber SI je zobrazený na obrázku Obr.2 – 2.

Po porovnaní ilustrácií oblastí záujmu I a SI je možné konštatovať, že tieto dve disciplíny majú veľa spoločného. Sústreďme sa teda na odlišnosti. Študenti SI sa v porovnaní so študentami I viac sústredia na spoľahlivosť softvéru i jeho údržbu. Ich pozornosť je tiež venovaná technikám a procesom spojených k činnostiam vývoja a údržby softvéru, ktoré sú korektné už v počiatočných fázach projektov. Študenti informatiky o týchto technikách a procesoch majú len základné poznatky a teda štúdijný program SI by v sebe mal zahŕňať vývoj softvéru, ktorý bude skutočne využívaný inými ľuďmi. Mať vedomosti a skúsenosti s dodaním reálne užitočného a použiteľného softvéru je vrcholne dôležité pre tento odbor.



Obr. 2 – 2 Oblasť záujmu softvérového inžinierstva.Prevzaté z (ACM & IEEE, 2013).

Vysoké školy na základe uvedených skutočností môžu nasledovať jeden z dvoch vzorov:

- zahrnúť do štúdijných osnov odboru I kurzy, ktoré pojednávajú o problematike softvérového inžinierstva v základoch, prípadne sa venovať kurzom SI prioritne ale stále v rámci odboru I,
- zahrnúť poznatky SI do samostatného študijného programu.

V súčasnosti je prvý bod predmetom horlivých diskusií, keďže oponenti takto zostaveného študijného programu poukazujú na rozdiel medzi vedcom a inžinierom (McConnell, 2004). Či sa už vysoká škola rozhodne pre prvý alebo druhý variant, podstatné je, aby v stredobode pozornosti bol študent a jeho potreby. Stotožňujúc sa s myšlienkami (Hislop, 2009) a (Lethbridge a kolektív, 2007) študijný program by mal mať nasledujúce atribúty:

- zatraktívnenie programu pre študentov,
- zamerať sa na menej všeobecnú výučbu ,

- efektívnejšie spolupracovať a komunikovať s priemyslom,
- vytvoriť nadčasové štúdijné osnovy,
- sprístupniť ďalšie vzdelávanie aj pracujúcim odborníkom,
- vedieť faktami podložiť vzdelanosť absolventov programu SI,
- zabezpečiť, aby lektori kurzov SI mali príslušné teoretické a praktické vedomosti,
- zvýšenie kvality a prestíže študentského výskumu v rámci softvérového inžinierstva.

Hľadaniu spôsobov, ktorými bude možné vytvoriť študijný odbor spomínaných kvalít sú venované ďalšie kapitoly.

3 Univerzitné prostredie stimulujúce študentov

„Hľadajme spôsob, aby učitelia menej učili a žiaci viac pochopili."
Jan Amos Komenský

Programovanie je v súčasnosti veľmi oceňovanou zručnosťou a môže priniesť skutočne zaujímavú kariéru a život na vysokej úrovni, avšak štatistiky (CS Education statistics, 2013) poukazujú na fakt, že zaznamenávame pokles záujmu študentov o informatiku. Univerzity, ktoré výučbu programovania zabezpečujú tak stoja pred neľahkou úlohou dostatočne motivovať študentov k učeniu sa a neodradiť ich od tvorby programov, keďže vysoké školy potrebujú každoročne pripraviť dostatočný počet absolventov pre prax a štúdijné programy, ktorých absolventi sú programátormi zaznamenávajú najväčší počet študentov vylúčených zo štúdia (A. Robins , J. Rountree , N. Rountree, 2003). V praxi sa stretávame s dvomi možnosťami prístupu k vyučovaniu programovania. Prvý je orientovaný smerom k softvérovému inžinierstvu, druhý k psychologicko-vzdelávaciemu smeru. Kolektív pána Robinsa sa vo svojej práci prikláňa k psychologicko-vzdelávaciemu prúdu, aj keď zdôrazňuje dôležitosť zahrnutia základov softvérového inžinierstva do výučby programovania.

Študenti sa teda vo všeobecnosti (aj na našej univerzite) dostanú k vývoju softvéru pomocou kurzov o programovaní. Individuálne riešia menšie, dobre zadefinované problémy s jasnou štruktúrou. Svoju pozornosť tak upriamujú na zvládnutie syntaxe a sémantiky daného jazyka za súčasného rozvoja svojich vedomostí v oblasti algoritmizácie. Študent sa v týchto predmetoch nedozvie o pozícii programovania v kontexte vývoja softvérového produktu. Až po zvládnutí týchto kurzov sa na rad dostávajú kurzy o softvérovom inžinierstve. Učia sa o životných cykloch programov, práce v tíme, tvorbe dokumentácie a podobne. Práve prvý kurz venovaný problematike SI býva najťažší, najmä z pohľadu pedagóga (Port D., Rachal C., Jia Liu, 2013). Lektori Havajskej univerzity mali podobné problémy. Ich študenti často komentovali absolvované kurzy výrokmi typu "Program viem vytvoriť aj bez tohto

predmetu!", "Všetky tieto procesné veci len oddiaľujú reálne vytváranie softvéru!", "Prečo musíme vypracovať také množstvo dokumentácie?". Nielen Havaj ale aj ostatné krajiny a ich univerzity musia na tieto výroky reagovať, aby študent pochopil, že *vývoj softvéru ďaleko presahuje aktivitu samotného programovania*, že práca v tíme si vyžaduje evidenciu aktuálneho stavu projektu a pochopiť svoj prínos k softvéru v celej dĺžke jeho existencie. Toto pochopenie je možné docieliť zmenou prístupu k vyučovaniu zavedením inovatívnych metód a odstránením nedostatkov zo štúdijných osnov, ktoré v roku 2011 Nurkkala a Brandle zosumarizovali do šiestich bodov (Nurkkala & Brandle, 2011):

- žiadny produkt - študenti vytvárajú školský projekt, nie komerčne použiteľný produkt,
- krátke trvanie kurzov - kurz v trvaní jedného semestra uvaľuje neprimerané časové obmedzenie na vývoj produktu,
- chýba rozvoj skúseností študenta v tejto oblasti informatiky založený na predchádzajúcich vedomostiach z prerekvizitných kurzov,
- nízka komplexnosť projektov kvôli časovému obmedzeniu,
- žiadna fáza údržby softvéru - táto kľúčová fáza v životnom cykle programu sa z časových dôvodov úplne vynecháva,
- absencia zákazníka - väčšina školských projektov nemá reálneho zákazníka.

Nasledujúce riadky tejto kapitoly sú venované prístupom, ktoré zvolili učitelia vysokých škôl, aby tieto nedostatky eliminovali, respektíve ich redukovali v rámci svojich možností.

3.1 Využívanie softvérových nástrojov vo výučbe SI

Bez ohľadu na konkrétny obsah kurzov softvérového inžinierstva, veľa lektorov si pri výučbe uvedomuje dôležitosť nástrojovej podpory a tomu prispôsobuje svoj prístup k výučbe. Používajú sa rôzne nástroje od plánovacích softvérov pri tvorbe programov, cez širokú škálu IDE až po konfiguračné nástroje. Študenti týmto spôsobom prídu do styku s nástrojmi, ktoré sa využívajú vo firmách a sú schopní s nimi efektívne pracovať, keďže poznajú ich silu aj slabiny.

Kolektív Akadémie vzdušných síl USA v Colorade v zložení (Teel, Schweitzer, Fulton, 2012) zaviedli do výučby SI otvorené nástroje pre zvýšenie kvality ich kurzu. Nástrojovú podporu, ktorú použili možno rozdeliť do štyroch kategórií: nástroje projektového manažmentu, nástroje projektovej komunikácie, programovacie nástroje a nástroje pre správu zdrojového kódu. Voľne dostupný nástroj Redmine bol autormi použitý ako nástroj projektového manažmentu i komunikácie. Medzi jeho výhody autori uvádzajú podporu viacerých projektov, dobrý systém sledovania úloh a teda kontrolu nad aktuálnym stavom rozpracovaného projektu, či už z pohľadu študentov v tíme alebo lektora. Systém podporuje aj komunikáciu vo vnútri samotného tímu, keďže k dispozícii je aj diskusné fórum pre členov projektu(viac detailov o komunikačných a manažérskych nástrojov je uvedených v prílohe C). IDE Eclipse bolo zvolené ako programovacie prostredie a Apache subversion bol použitý ako nástroj pre správu zdrojového kódu. Dlhodobejšie výsledky z tohto prístupu zatiaľ k dispozícii nie sú, avšak univerzita konštatuje zlepšenie výsledkov. Študenti pochopili prínos nástroja pre projektový manažment ako prostriedok efektívnej komunikácie a mnohí ocenili zavedenie správy zdrojového kódu do takej miery, že si použitie týchto nástrojov vyžiadali aj v iných kurzoch. Autori dospeli k nasledovným finálnym záverom. Používanie štandardných vývojárskych nástrojov a vývojárskych prostredí vyúsťuje v efektivite práce tímu i lektora, ktorý má lepšiu kontrolu nad správou študentov ako aj administráciou celého systému študentských projektov. Podobne,

komunikácia v rámci tímu sa zlepšila, komunikačné uzly sú konzistentné, stále dostupné a nezávislé od konkrétnej lokácie členov tímu. Tretiu výhodu autori vidia v dostupnosti mentora v akomkoľvek čase, stráca sa nutnosť konzultačných hodín. Poslednou podstatnou výhodou takto navrhnutej organizácie kurzu je simulácia pracovného prostredia, do ktorého väčšina študentov tohto kurzu vstúpi po úspešnom absolvovaní štúdia (Teel, Schweitzer, Fulton, 2012). Práca spomínaného kolektívu predstavuje sumár snáh o zaradenie nástrojovej podpory do procesu výučby, ktorý si osvojili viacerí lektori SI pred nimi aj po nich. Osobité prínosy predošlých pokusov sú uvedené ďalej v tejto časti práce, kvôli demonštrácii potreby neustáleho pretvárania výučby pri objavení sa stále dokonalejších a voľne dostupných nástrojov pre tvorbu softvérových produktov.

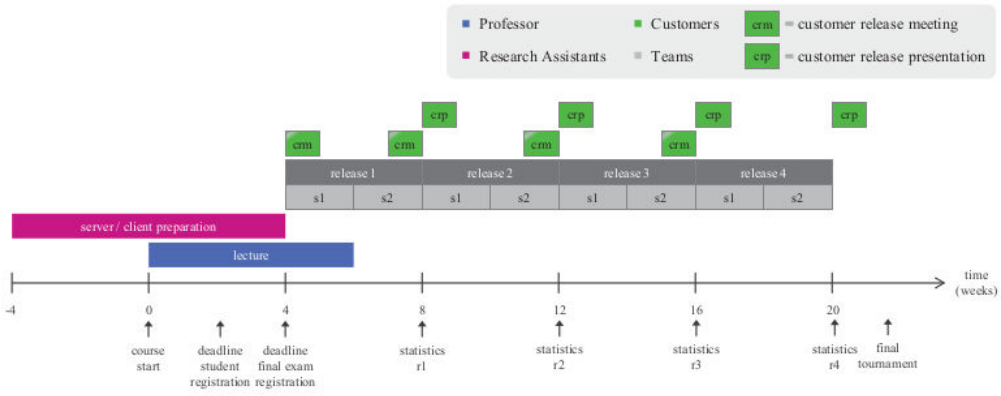
(Shihong, Distante, 2006) motivujú pri výučbe SI študentov tým, že ich nechávajú vybrať si zadanie projektu výlučne samostatne. Veria, že takto zvýšia ich zanietenosť pre projekt. V jednotlivých fázach tvorby programu majú možnosť pracovať so známymi pojmami a sústrediť sa na procesy pri tvorbe softvéru, nie na doménu problému. Využitý bol jediný komplexný nástroj od firmy IBM pod názvom Rational Suite Enterprise, masívny platený nástroj zahŕňajúci všetky aspekty tvorby programov, od kolekcie požiadaviek až po správu konfigurácií.

Nástroje pre podporu vytvárania si univerzity vytvárajú aj samé, aby si svoj softvér ušili na mieru požiadavkám daného predmetu. Podobne postupovali odborníci na výučbu SI v Nemecku. (Gnatz, Kof, Prilmeier, Seifert, 2003) Ich nástroj APE (Applied Patterns Environment) umožňuje v procese vývoja produktu postupovať podľa procesných vzorov, keďže APE navrhne vhodný procesný vzor na základe stavu dokumentov projektu. Napríklad ak je dokument prípadov použitia označený ako akceptovaný, tak program navrhne vzor testov pre prípady použitia.

3.2 Zavedenie SCRUM do výučby SI

Metodológia SCRUM sa zaraďuje medzi agilné metodológie, v ktorej vývoj softvéru prebieha vo viacerých iteráciách. Je zameraná na spoluprácu a komunikáciu, funkčnosť produktu a v neposlednom rade na flexibilitu produktu tak, aby bol adaptabilný požiadavkám používateľov (SCRUM, 2013). Agilné metodológie vo všeobecnosti podporujú prácu v tíme, čo z nich robí vhodného kandidáta pre účely tejto práce, ktorá sa snaží vychovať softvérových inžinierov pripravených pre prax. Výber SCRUM nie je náhodný, pretože podľa (Survey, 2012) je to najpoužívanejšia metodológia v praxi.

Podobné pohnútky viedli aj pánov Kocha a Scharfa k zavedeniu tejto metodológie do výučby. Druhý dôvod, ktorý autori spomínajú je fakt, že časovanie metodológie SCRUM je v symbióze s dĺžkou jedného semestra na ich univerzite, konkrétne 16 týždňov. Dĺžka semestra je rozdelená na štyri dvojtýždňové obdobia, v ktorých prebiehajú dva šprinty končiace vydaním verzie, tak ako to ilustruje obrázok (Obr. 3 – 1). Je možné vidieť, že pred prvým šprintom je potrebná ďalšia 4-týždňová teoretická príprava.



Obr. 3 – 1 Priebeh štúdia kurzu SI v Nemecku. Prevzaté z (Scharf, Koch, 2013).

Celý kurz prebieha nasledujúco. Študent sa prihlási do kurzu, vyznačí práve jedného

kolegu, s ktorým chce pracovať a je mu náhodne pridelený zvyšný tím(náhodnosť je podmienená predchádzajúcimi skúsenosťami, kedy vznikali veľké rozdiely v kvalite tímov). Každému tímu sú pridelení študenti doktorandského štúdia, ktorí sú mentormi týchto skupín a majú s nimi povinné stretnutia pred a po vydaní verzie. Tu vyvstáva problém s dostatočným počtom študentov tretieho stupňa vysokých škôl, avšak autori nechcú z tohto počtu poľaviť kvôli autenticite výučby. Praxou sa ukázalo, že žiaci potrebujú pomoc najmä počas prvého dvojtýždňového obdobia, kedy reálne po prvýkrát prichádzajú do styku s technológiami Agilo a metodológiou SCRUM ako takou.

Ďalšími dvomi doktorandami sa zaplnia dve pozície zákazníkov, ktorí sa študentom predstavia v druhom alebo treťom týždni semestra na prednáške. Prezentujú svoje požiadavky na softvér a demonštrujú nástroje, ktoré používajú - konkrétne ich blog pre komunikáciu a Google kalendár. Tieto nástroje musia využívať aj tímy študentov, aby vedeli dodržiavať termíny zákazníkov. Pre zvýšenie dôveryhodnosti sú zákazníci majiteľmi fiktívnej firmy. Následne sa vykonávajú jednotlivé šprinty, stretnutia a vydávanie verzií konkrétnych produktov. Autori konštatujú spokojnosť študentov i učiteľov s takto organizovaným kurzom softvérového inžinierstva (Scharf, Koch, 2013).

Trocha odlišný prístup rozdelenia času bol zavedený na univerzite v Amsterdame. Predmet bol venovaný výuke softvérového dizajnu a univerzita sa pokúsila o zavedenie spoločenských stretnutí. Konkrétna realizácia prebiehala v týždňových cykloch, kde v prvý deň týždňa sa konala prednáška, tretí deň bol venovaný revízii od rovného (peer review) a v posledný, piaty deň týždňa patril stretnutiu so zákazníkom. Práve tretí a piaty deň v sebe nesú podstatu sociálneho stretávania sa medzi študentami a lektormi. (Tamburri, Razavian, Lago, 2013) vykonávajú revíziu rovného inteligentným spôsobom. Každý týždeň pred tretím dňom je jednotlivým tímom rozposlaná rozpracovaná verzia zadania konkrétneho týždňa. Nasledujúci deň každý tím vyhodnotí zadania všetkých ostatných tímov a pošle im svoje analýzy. Študenti

tak získajú prehľad o napredovaní ostatných tímov, učia sa z vlastných chýb aj chýb iných. Podstatná je aj komunikácia v piaty deň týždňa, kedy sa diskutuje o zadaní so zákazníkom. Spoločným dialógom sa dosiahne konsenzus a na ďalší týždeň sa produkt vyvíja podľa ďalších požiadaviek.

Princíp revízií od rovného sa využíva aj v iných predmetoch zaoberajúcich sa SI. Vo svojej podstate ide o typ revízie, pri ktorej ľubovoľný produkt(špecifikácia požiadaviek, dokument návrhu, zdrojový kód a pod.) je skontrolovaný samotným autorom a rovnako aj prinajmenšom jedným kolegom pre dosiahnutie čo najlepšej kvality(Wiegers, 2002). (Garousi, 2010) vo svojej práci uskutočnil v dvoch po sebe nasledujúcich rokoch experiment s takýmto spôsobom revízie. Využíval menej formálnu formu, ktorá bola pre študentov prirodzenejšia. Garousi najpozitívnejšie vníma pokrok k lepšiemu u študentov pri pripravovaní kvalitných dokumentov návrhu a podobne aj pri identifikácii chýb vo svojej alebo cudzej práci.

3.3 Ďalšie aktivity smerujúce k vyššej kvalite výučby SI

Myšlienka presunutia tradičnej prednášky z priestoru auly do prostredia internetu nie je nová. Zástancovia invertovaných tried argumentujú tým, že učiteľ počas svojej prednášky sa môže venovať iným, možno zaujímavejším aktivitám. (Herold a kolektív, 2012) sa venoval aplikovaniu tejto myšlienky do praktickej výučby SI. Tvrdí, že prednášajúci môže počas hodiny interaktívne demonštrovať príklady, komunikovať s poslucháčmi, ba dokonca venovať sa workshopom. Online prednášky by tak zostali na doplnenie učiva v rámci samoštúdia. Do popredia sa dostáva otázka, či takto navrhnuté osnovy nie sú príliš veľkým sústom pre študenta, ktorý musí zvládať aj štúdium ďalších kurzov. Z výsledkov autora spomínanej štúdie vyplýva, že študenti nepociťovali radikálne väčšiu záťaž z takto organizovaného kurzu. Práve naopak, nudné teoretické prednášky si pozreli v domácom prostredí, kde sa mohli rozptýliť a počas prednášok mali možnosť diskutovať o praktických záležitostiach, avšak

nezabudol podotknúť fakt, že kvalita týchto diskusií záležala na úrovni prítomných študentov.

Zaujímavým spestrením hodín softvérového inžinierstva je aj počin s ktorým prišla skupina okolo pána Zhu-a (Qing Zhu, Tao Wang, Shenglong Tan, 2007). Do výučby vložili simulačné programy, ktoré mali tímom uľahčiť prácu. Išlo o programy SimSE a MO-SEProcess. SimSE je edukačná 2D hra, ktorej úlohou je premostenie medzery medzi veľkým množstvom učiva preberaného na prednáškach a malým praktickým vyskúšaním nadobudnutých vedomostí v praxi. Študent - hráč - sa ocitá v roli projektového manažéra, ktorý má na starosti projekt strednej veľkosti a zložitosti. Jeho úlohou je postupne zadeľovať úlohy šiestim softvérovým inžinierom podľa ich schopností a následne tieto úlohy upravovať podľa náhodných udalostí, ktoré sa vyskytnú počas hry. Cieľom je vydať verziu programu, ktorá bude v čo najpokročilejšom štádiu. Ak hráč nedodá program do stanoveného času, prípadne ak náklady stúpnu, hráč hru prehral.

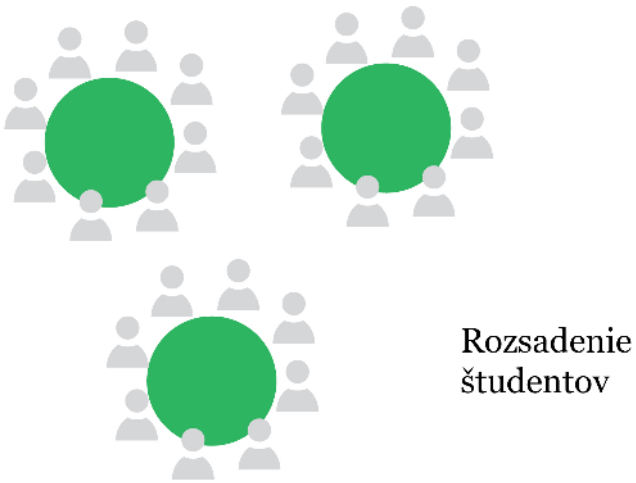
MO-SEProcess vychádza zo SimSE. Ide o 3D hru, v ktorej hráč spolupracuje s inými hráčmi s cieľom vytvoriť softvérový systém. Používatelia manažujú avatarov dávajúc im rôzne úlohy. Tí ich plnia (v prípade, že nemajú pracovnú prestávku) podľa svojich schopností a zručností. Podporovaný je zatiaľ len vodopádový model. Každý z hráčov(maximálne šesť) zapojených do hry sa stáva členom tímu. Môže si vybrať ľubovoľnú rolu a potom má na starosti plnenie jej úloh. V prípade úspešnej implementácie je na konci hry tímu udelené príslušné skóre.

V zimnom semestri roku 2012 sa pokúsila Štátna univerzita v indickom hlavnom meste o vypracovanie a vyhodnotenie nového prístupu k výučbe softvérového inžinierstva v bakalárskom stupni štúdia. Cieľom štvorice lektorov: Ashish Sureka, Monika Gupta, Dipto Sarkar a Vidushi Chaudhary bolo naučiť základy SI pomocou aktívne-ho učenia značným zvýšením spolupráce na úrovni študent-inštruktor a študent-študent, tímovej spolupráce, rovnováhou medzi teóriou a praxou v me-

dziach technických i riadiacich. (Sureka, Gupta, Sarkar, Chaudhary, 2007) vystavali osnovu pre svoj kurz na štyroch pilieroch: invertovaná trieda, štúdiovo-orientované učenie, projekty pre reálnych klientov, veľké tímy s revíziou od rovného.

Kvôli kolaboratívnej povahe predmetu boli všetci študenti zadelení do relatívne veľkých skupín v rozmedzí 8 až 9 študentov. Jednotlivé skupiny si využitím Wiki Tab vytvorili vlastné Wiki stránky pre ich projekty a pre dokumentáciu projektov použili Google Project Hosting Wiki. Počas vytvárania softvéru sa študenti v rámci tímu ohodnocovali sami pomocou revízií od rovných, kde mali objektívne hodnotiť účasť jednotlivcov na danom projekte. V anonymnom dotazníku sa však študenti sťažovali na manipuláciu týchto údajov.

Druhým cieľom projektu bolo vytvorenie netriviálnych softvérov pre reálneho zákazníka. Inštitút si za zákazníka vybral sám seba a študenti riešili otázky týkajúce sa zlepšenia tejto organizácie. Kvôli simulácii reálneho sveta sa zlepšenia netýkali katedry informatiky ale oddeleniam personalistiky a ľudských zdrojov, financií a podobne. Počas hodín v škole boli študenti v triede usadzovaní v modeli štúdiovo-orientovanom, ktorý je ilustrovaný na obrázku Obr.3 – 2. Tento model podporuje



Obr. 3 – 2 štúdiovo-orientovaný model. Prevzaté z (Sureka, Gupta, Sarkar, Chaudhary, 2007).

lepšiu spoluprácu už na úrovni školského semináru.

Kvôli organizácii cvičení formou štúdiovo-založeného učenia nebolo možné dostatočne sa venovať klasickým prednáškam. Z tohto dôvodu sa prednášky konali v online forme v osobnom voľne študentov.

4 Inovatívna výučba predmetov o programovaní

„Umenie začína imitáciou a končí inováciou."
Mason Cooley

Hľadanie spôsobu podania nových poznatkov študentom, zvýšenie ich pozornosti a zájmu o programovanie ako aj uľahčenie práce pre pedagógov sú náplňou tejto práce. Ako to už ale býva, myšlienka, ktorá príde na um Vám, zväčša napadne aj niekomu inému. Vďaka tejto skutočnosti ani táto práca nemusí začínať úplne od nuly, bez existujúcich výsledkov tuzemských alebo zahraničných kolegov, jednoducho sa môžeme oprieť a imitovať prístupy skúsenejších. A že sa máme kde inšpirovať bolo dokázané v predchádzajúcej kapitole. Poďme sa teda pozrieť na možnosti ako kurzy o programovaní zmeniť tak, aby boli v súlade s cieľmi tejto práce. Inšpirujeme sa pri tom existujúcimi prístupmi, ku ktorým pridáme vlastné obohatenie v zmysle citátu pána Cooleyho.

Uvedomujúc si súčasný stav možností štúdia informatických vied na našej univerzite, zvolíme si možnosť výučby procesov softvérového inžinierstva v predmetoch o programovaní. Kvôli zjednodušeniu a experimentálnej povahe práce, naše snahy o zlepšenie výučby budú namierené na jediný kurz, konkrétne kurz Technológie JAVA, ktorý je v osnovách druhého ročníka bakalárskeho stupňa štúdia. Pri inovácii sa špecializujeme na zmenu formy výučby cvičení predmetu, nie na zmenu ich obsahovej stránky, ktorú však pre úplnosť táto práca neopomenie.

Obsahom cvičení spomínaného predmetu je prípadová štúdia Minesweeper. (Jianmin Zhang, Jian Li, 2010) tvrdí, že prípadová štúdia je vhodná ako metóda pre výučbu programovania, keďže študent postupne aplikuje poznatky z prednášok do vývoja projektu štúdie a vidí súvislosti lepšie v porovnaní s programovaním separátnych algoritmov, ktoré im nedávajú hlbší význam. Aj z tohto dôvodu nie je potrebné meniť obsah cvičení. Koniec-koncov, študent na konci semestra vyvinie

funkčnú verziu populárnej hry Míny, pomocou ktorej zvládne základy technológií JAVA od zapísania jednoduchých algoritmov v rovnomennom jazyku až po prácu s databázami pri ukladaní dosiahnutých výsledkov počas hrania hry. Mnoho študentov sa pre programovanie rozhodne práve vďaka hraniu hier a vytvorenie prvej vlastnej hry mu prináša skutočnú radosť, preto priestor na zlepšenie kurzu hľadajme vo forme cvičení.

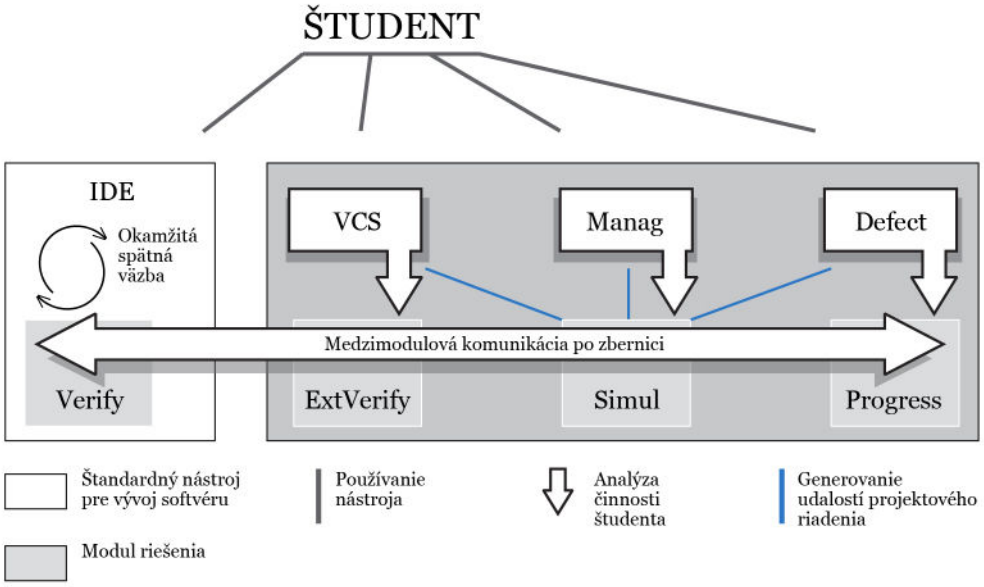
Cvičenia predmetu sú organizované nasledovným spôsobom. V priebehu semestra je každému študentovi k dispozícii dvanásť cvičení v trvaní 90 minút, počas ktorých postupne implementuje svoj projekt. Na každom stretnutí je študentovi daný kompletný návod ku konkrétnemu týždňu obsahujúci ciele daného týždňa ako aj zoznam úloh s poznámkami. Lektor na začiatku hodiny tieto ciele svojim žiakom ozrejmí, vysvetlí prípadné nejasnosti a následne prichádza na rad samostatná práca študenta pod dohľadom lektora. Opísaný prístup je tradičným školským pohľadom na výučbu, keď aplikácii poznatkov do praxe predchádza vysvetlenie problému. Zaužívaný školský prístup ale v sebe implicitne obsahuje tradičné nedostatky.

- Študentovi je cieľ hodiny predostretý ako nemenná entita, o ktorej nie je potrebné diskutovať, teda študent nepristupuje k zadaniu kriticky.
- Učebný text je pripravený ako úloha pre celý týždeň na jednej webovej stránke, čím sa stráca interaktivita medzi lektormi a študentami. Študent plní úlohy v texte, bez komunikácie a spätnej väzby od odborníkov.
- Lektor zasahuje do študentovho projektu prevažne až vtedy, keď je vyzvaný od študenta avšak z praxe je známe, že študent s otázkou vyčkáva kvôli obavám či ostychu.
- Kontrola postupu zadania nie je priebežná, teda vyučujúci pri veľkom počte študentov sa k niektorým študentom dostane až pri samotnom odovzdávaní projektu a daný študent tak môže mať oprávnený pocit, že je ignorovaný.

Takto jasne špecifikované nedostatky výučby ponúkajú vhodnú východiskovú situáciu pre vytvorenie lepších štúdijných podmienok. Skúsme si predstaviť, že študenta vymaníme z kolobehu deväťdesiat minútových rozvrhových jednotiek v priebehu dvanástich týždňov a to tak, že sa stane aktívnym členom tímu zaoberajúceho sa vývojom aplikácie. Poslucháč tak bude postavený do úlohy zamestnanca firmy, čo implikuje následovné. Autonómne riešenie študenta bude konfrontované s požiadavkami na daný softvér prostredníctvom kontroly skúsenejších členov tímu. Členovia tímu tak odovzdajú študentovi veľmi cennú spätnú väzbu, z ktorej sa môže poučiť, prípadne sa študentova aplikácia modifikuje pridaním kódu, keďže v tíme vývojárov sa zdrojový kód aplikácie rodí spoluprácou. Interakcia tohto druhu sa stane prospešnou pre študenta a školu posunieme do role inštitúcie, ktorá žiakovi dá potrebné informácie, ktoré sú pre jeho prácu nevyhnutné, keďže ich hneď implementuje. Poznatky členov tímu reprezentujú praktické skúsenosti s aplikovaním týchto vedomostí v konkrétnych príkladoch.

Po abstraktom predstavení koncepcie riešenia sa pozrime na jej bližšie špecifikovanie definovaním obmedzení, ktoré existujú na našej univerzite. Najväčším problémom je spolupráca s externou firmou, ktorá by vedela uspokojiť požiadavku fiktívneho zamestnania študentov a vyčleniť zamestnancov pre komunikáciu s nimi. Je preto nevyhnutné pozrieť sa na virtuálne riešenie, kedy si vytvoríme celý rad spolupracujúcich celkov, ktoré vyústia do harmonického celku podporných nástrojov vyučovacieho procesu (Obr. 4 – 1). Obrázok (Obr. 4 – 1) rozdeľuje systém do dvoch základných, samostatných ale spolupracujúcich, celkov. Prvý celok predstavuje prácu študenta, ktorý v integrovanom vývojovom prostredí(IDE) realizuje priradené zadanie. V rámci tohto celku je študentovi pri kompilácii programu poskytnutá okamžitá spätná väzba od modulu *Verify*, ktorý je súčasťou IDE. Modul vyhodnotí syntaktickú a sémantickú korektnosť programu, čím podá študentovi výpovednejšie informácie ako samotný prekladač.

Druhá časť systému je venovaná réžii cvičení, konkrétne pojednáva o možnostiach



Obr. 4 – 1 Konceptuálny návrh nástrojovej podpory pre výučbu procesov softvérového inžinierstva

kontroly študentom odoslaného zadania, vyvodení následkov tejto kontroly a vyhodnotení postupu žiaka. Celý postup je možné opísať v nasledovnom poradí. Študent odošle svoje zadanie do systému pre správu verzií(VCS), v ktorom modul *ExtVerify* vyhodnotí prijaté riešenie. Kontrola spočíva v opätovnej analýze korektnej syntaxe a sémantiky programu, ako aj v možnosti kontroly originality. Výsledky sú odoslané modulu *Simul*, ktorý je zodpovedný za životaschopnosť celého druhého celku. Na základe výsledkov vyhodnotí stav projektu daného študenta a rozhodne o pridelení novej úlohy, respektíve opätovne vyzve študenta k implementácii rovnakého problému v prípade, že modul *ExtVerify* vyhodnotil aktuálne riešenie ako nesprávne. Rozhodnutia simulačného modulu sa prejavia v systéme pre projektové plánovanie, riadenie a evidenciu výkonov(Manag) ako aj formou elektronickej pošty. Posledným modulom je *Progress*, ktorý prislúcha nástroju pre evidenciu chýb riešenia študenta(Defect), kde sa evidujú chyby projektu a na základe týchto údajov

je možné vyhodnotiť postup študenta. Diplomová práca, ktorú čitateľ drží v rukách sa zaoberá simulačným modulom v systéme nástrojovej podpory pre integrovanú výučbu procesov softvérového inžinierstva v kontexte programovania s okamžitou spätnou väzbou.

4.1 Potreba externého manažéra v procese výučby

Najpodstatnejšou úlohou simulačného modulu je vytvorenie simulácie procesov reálne prebiehajúcich vo firmách, kedy sa študent ocitne v pozícii vývojára aplikácií. Z toho je zrejmé, že potrebujeme do systému priviesť osoby zodpovedné za ďalšie úlohy, ktoré sa vyskytujú v praxi. Minimálne je potrebný človek, ktorý bude prideľovať študentovi úlohy. Je zrejmé, že túto úlohu by mohol vykonávať cvičiaci alebo prednášajúci daného predmetu, avšak to by bolo na škodu celému výskumu, keďže simulovať chceme práve procesy prítomné vo firmách a školu posunúť do roviny poskytovateľa poznatkov nevyhnutných pre ďalšie napredovanie v budúcom kariér-nom živote.

Ďalšou výhodou, ktorú poskytne postava manažéra vyplýva z požiadavky pre dopĺňanie kódu študentov. V procese tvorby hry Míny totiž študent implementuje časti kódu, ktoré v neskorších etapách nebude potrebovať, prípadne kvôli otestovaniu študenta z porozumenia kódu, budú vložené chybné riadky zdrojového kódu. Takéto správanie je vo firmách častým javom, pretože objednávateľ produktu mení svoje požiadavky a rovnako aj kolegovia v tíme môžu urobiť pri programovaní chyby. Takéto cielené robenie chýb je ale v rozpore s etickým kódexom učiteľa.(Slovenská komora učiteľov , 2010) hovorí, že učiteľ má podávať fakty a informácie pravdivé, objektívne a neskreslené a to bez ohľadu na svoje vlastné presvedčenie. Rovnako, lektor nesmie žiakovi úmyselne či vedome ublížiť.

4.2 Potreba vytvorenia rozsiahleho tímu v procese výučby

Simulácia sa stane reálnejšou za predpokladu, že sa študent ocitne v prostredí, ktoré je uveriteľné. Z toho vyplýva, že entita manažéra nebude postačujúca. Len s problémami je možné si predstaviť manažéra firmy, ktorý komunikuje zo zákazníkom, následne požiadavky zapracuváva do úloh pre programátora, vyriešené úlohy kontroluje a posiela vývojárom spätnú väzbu. Pre potreby výskumu je ale potrebné, aby sme podmienky realizácie cvičení predmetu priblížili čo najbližšie k realite.

Je preto dôležité vytvoriť aspoň dve virtuálne úlohy. Prvou bude už spomínaný manažér. Komunikácia s touto postavou v systéme bude sporadickejšia a pojednávať bude o záležitostiach manažérskeho charakteru, konkrétne požiadavky zákazníka, termíny odovzdania riešenia úloh a podobne. Druhá úloha bude venovaná postave skúsenejšieho vývojára aplikácií. Práve od tejto postavy dostane študent spätnú väzbu k svojej práci, pochvalu i pokarhanie, jednoducho taký virtuálny poradca. V ideálnom prípade sa nám podarí vytvoriť celý tím, ktorý bude pozostávať z manažéra a viacerých skúsenejších programátorov, aby sa tak v rámci cvičení študenti organizovali do viacerých skupín, kde každej skupine bude priradený niektorý z tímu virtuálnych programátorov. Títo programátori budú mať svoj psychologický profil tak, aby simulácia bola dôveryhodná. Ich povahy by tak zodpovedali rozsahu pováh od flegmatikov nerešpektujúcich termíny až po poctivých workoholikov. Vytvoreniu osobnostných profilov virtuálnych členov tímu na základe princípov kognitívnej psychológie je venovaná nasledujúca kapitola.

5 Odpoveď psychológie na modelovanie osobnosti

„Mnoho ľudí sa domnieva, že odborníka
robí odborníkom jeho intelekt.
Mýlia sa. Je to jeho osobnosť."
Albert Einstein

Vytvorenie modelu, ktorý by uspokojivo vyjadril ľudskú osobnosť v celej jej komplexnosti a teda aj jej rôzne anomálie bolo jedným zo základných cieľov psychológie. Zobrazenie osobnosti do modelu by umožnilo zlepšenie všeobecného pochopenia ľudského charakteru a prevencie osobnostných porúch. Odborníci vytvorili viacero modelov, z ktorých sa jeden v súčasnej psychológii teší veľkej obľube. Pre tento model je príznačné využitie piatich základných vlastností alebo dimenzií osobnosti, preto sa mu v anglicky hovoriacich krajinách dostalo označenia *Big 5 model*. (Digman, 1990) považuje model piatich vlastností za vysoko praktický a aplikovateľný v oblasti psychológie osobnosti.

Medzi spomínané faktory zaraďujeme *Openess to experience* (O), *Conscientiousness* (C), *Extroversion* (E), *Agreeableness* (A) a napokon *Neuroticism* (N) (Popkins, 1998). Zložením začiatočných písmen týchto slov získame akromyn OCEAN, ktorý sa taktiež používa pre označenie modelu piatich faktorov. Bližšie sa pozrime na jednotlivé vlastnosti tak, ako sa prejavujú v charaktere človeka a ako ich definoval (Beaumont, 2003).

Vlastnosť otvorenosti k zážitkom a skúsenostiam (O) rozdeľuje ľudí na tvorivých s dobrou predstavivosťou a konvenčných jedincov s nohami na zemi. Otvorení ľudia sú zvedaví, zaujíma ich umenie a krása. Protipólom sú ľudia, ktorí uprednostňujú jednoduchosť, priamočiarosť a jasnosť oproti komplexnému alebo vágnemu prístupu.

Svedomitosť (C) je vlastnosť, ktorej záujmom je spôsob, akým ľudia kontrolujú svoje impulzy. Impulzy nemusia byť nevyhnutne negatívne, napríklad konanie podľa pr-

votných impulzov v časovo náročných podmienkach sa môže ukázať ako veľmi výhodné, v prípade, že človek vie pracovať pod nátlakom. Táto vlastnosť v sebe zahŕňa aj túžbu človeka niečo dosiahnuť. Ľudia s touto vlastnosťou sa snažia vyhýbať sa problémom a dosiahnuť úspech za pomoci efektívneho plánovania a vytrvalosti. Inými jedincami sú považovaní za inteligentných a spoľahlivých. Na druhej strane, ľudia s touto vlastnosťou sú perfekcionisti a workoholici. Nespoľahlivosť, nedostatok ambícií a nerešpektovanie pravidiel sú charakteristické povahové črty ľudí so zníženou úrovňou svedomitosti.

Extroverzia (E) charakterizuje prepojenie jedinca s vonkajším svetom. Extroverti sa radi zdržiavajú v spoločnosti, sú plní energie a zvyknú pociťovať pozitívne emócie. Sú to ľudia aktívni, entuziastickí, v spoločnosti zhovorčiví s tendenciou upriamiť pozornosť na seba. V opozícii ku extrovertom stoja introverti, teda ľudia oddeľujúci sa od spoločnosti iných, avšak nedostatok spoločenského života nie je potrebné interpretovať ako hanblivosť alebo depresie.

Vlastnosť ochoty alebo sympatickosti (A) odráža postoj k spolupráci a spoločenskej harmónii. Ľudia s prevahou tejto vlastnosti majú zväčša dobré vzťahy s ostatnými, sú priateľskí, nápomocní a prístupní kompromisom. Taktiež sú to ľudia dôverčiví. Jedinci s nedostatkom tejto vlastnosti uprednostňujú svoje potreby pred ostatnými, často sú nedôverčiví, podozrievaví, odmeraní a neradi spolupracujú.

Posledná vlastnosť, neurotickosť (N), pojednáva o tendencii jedinca prežívať negatívne pocity. Vysoko neurotický ľudia často pociťujú hnev, depresie a podobne. Prežívajú veľmi emotívne aj také udalosti, ktoré ostatní prehliadajú, menšie problémy sa im kvôli ich schopnosti preháňať zdajú ako neprekonateľné prekážky. Negatívne pocity sa v nich ukladajú na dlhší čas a spôsobujú ich podráždenosť i zlú náladu, čo má za následok zníženú kvalitu objektívneho myslenia, tvorby rozhodnutí a práce pod stresom. Opakom sú ľudia, ktorí sú citovo stabilní a menej náklonní k trvalým negatívnym pocitom.

Spomínané vlastnosti predstavujú len základnú množinu charakteristík, podľa ktorých je možné definovať osobnosť jedinca. Je to však abstraktný model a kvôli komplexnosti ľudskej povahy mu nedovoľuje zachytiť všetky črty ľudskej povahy. (Cherry, 2008) uvádza medzi hlavné nezahrnuté aspekty osobnosti:

- manipulatívnosť,
- čestnosť,
- pohlavie,
- vierovyznanie,
- snobizmus,
- konzervativizmus.

Za veľkú slabinu tohto modelu sa považuje aj fakt, že nie je podporený žiadnym teoretickým základom. Ide o model vytvorený na základe pozorovaní a nie sú teoreticky preskúmané všetky prepojenia, ktoré existujú medzi piatimi východiskovými vlastnosťami.

Nevýhoda pre teoretických psychológov sa môže ukázať ako výhoda pre prax, keďže model OCEAN vychádza práve z nej. Ide o model, ktorý popisuje vlastnosti ľudí a vie jednotlivcov zatriediť do rôznych skupín. Je natoľko flexibilný, že ho možno aplikovať do rôznych oblastí života od výberu vhodného kandidáta do práce (Lucius, 2003) až po vytvorenie virtuálnych osobností pre potreby výučby programovania.

5.1 Využitie modelu OCEAN vo výučbe predmetov o programovaní

Kvôli vysokej flexibilite modelu OCEAN si päť základných faktorov pre definíciu osobnosti človeka môžme prispôsobiť podľa vlastných potrieb. Vieme, že na cvičeniach kurzu *Technológie JAVA* študentom budú prideľované úlohy s termínmi ich splnenia ako aj extra úlohy pre lepšie motivovanie študentov. Jednotlivé úlohy im budú prideľované automatizovaným systémom, ktorý ich bude aj hodnotiť, teda pôjde o virtuálne osoby s prispôsobeným výkladom vlastností.

- Vlastnosť O:
  - znamená otvorenosť voči novej skúsenosti, otvorenosť voči nápadom alebo tiež intelekt,
  - hovorí o vzťahu k inovácii, k novým myšlienkam,
  - predstavuje zvedavosť, kreativitu a široké spektrum záujmov,
  - nadobúda hodnoty od 0 po 1, 0 pre absenciu vlastnosti, 1 pre maximálnu mieru vlastnosti,
  - hodnota 0 predstavuje uzavretosť mysle, hraničnú konzervatívnosť, neakceptovanie akýchkoľvek odchýlok od tradičných postupov,
  - hodnota 0,5 je rovnováhou, oceňuje rovnako originalitu ako aj preverené postupy a myšlienky, nevyhľadáva prioritne a neprekáža mu ani jeden ani druhý postoj,
  - hodnota 1 predstavuje otvorenosť mysle, neustálu snahu o inováciu aj o vlastné postupy, odmieta tradičnosť a snaží sa dosiahnuť vlastné riešenia, oceňuje originalitu iných.

- Vlastnosť C:
  - znamená spoľahlivosť, organizovanosť, zodpovednosť, sebadisciplínu,
  - predstavuje plánovanie, zameranie sa na úspech,
  - dáva do pomeru efektívnosť/organizovanosť proti ľahostajnosti/spontánnosti,
  - nadobúda hodnoty od 0 po 1, 0 pre absenciu vlastnosti, 1 pre maximálnu mieru vlastnosti,
  - hodnota 0 predstavuje neorganizovanosť, nikdy nič neplánuje dopredu a je neambiciózny, nevyžaduje od ostatných dochvíľnosť, nemá zmysel pre žiadnu systematickosť, nízka pracovná morálka,
  - hodnota 0,5 je rovnováhou, prikladá primeraný dôraz na organizáciu práce, organizuje len do určitej miery, dáva priestor aj spontánnosti, má priemernú pracovnú morálku,
  - hodnota 1 - dodržiavanie pravidiel a cieľov je prioritou, kladie vysoký význam na časovú organizovanosť, neznesie nepresnosť a lenivosť, vysoká pracovná morálka.
- Vlastnosť E:
  - znamená energickosť, spoločenskosť, zhovorčivosť, asertivitu,
  - vystihuje spôsob prejavu,
  - dáva do pomeru extroverziu/spoločenskosť naproti introverzii/uzavretosti,
  - nadobúda hodnoty od 0 po 1, 0 pre absenciu vlastnosti, 1 pre maximálnu mieru vlastnosti,

  - hodnota 0 predstavuje uzavretú osobnosť, spoločnosť ostatných mu nechýba a často aj prekáža, samotár, preferuje pracovať sám, úzky okruh priateľov, stručný a nevýrečný spôsob komunikácie a vyjadrovania,
  - hodnota 0,5 je rovnováhou, nepreferuje samotu ani spoločnosť, neprekáža mu ani jedno ani druhé, primerane spoločenský, vie pracovať aj sám, primeraný spôsob komunikácie,
  - hodnota 1 - spoločenský, obklopuje sa ľuďmi, neznesie samotu, uprednostňuje prácu v kolektíve, komunikuje bohato, preferuje dlhé slovné spojenia, veľa rozpráva.

- Vlastnosť A:
  - znamená priateľskosť, súcit, pochopenie,
  - vystihuje schopnosť pracovať v kolektíve,
  - dáva do pomeru priateľskosť/ zmysel pre pochopenie naproti profesionálnosti/osobnej chladnosti,
  - nadobúda hodnoty od 0 po 1, 0 pre absenciu vlastnosti, 1 pre maximálnu mieru vlastnosti,
  - hodnota 0 predstavuje človeka, ktorý je neosobný, chladný, nikdy nenadväzuje priateľské vzťahy, pokiaľ nemusí, vo vedúcej pozícii kritizuje každú chybu, netoleruje neschopnosť, preferuje profesionálny spôsob práce a vedenia, neodpúšťa a je nekompromisný,
  - hodnota 0,5 je rovnováhou profesionálnosti a ľudského prístupu, vie pochopiť a odpustiť, ale len v určitej miere, vyžaduje primeranú kvalitu,
  - hodnota 1 - súcitný, priateľský, osobný, ľahko sa zneužíva, lebo je dôverčivý, neprikladá veľký dôraz na kvalitu, zaujímajú ho iné črty.

- Vlastnosť N:
  - znamená emocionálnu nestabilitu,
  - vystihuje slabú odolnosť voči stresu, psychickému tlaku, impulzívnosť,
  - dáva do pomeru emocionálnu nestabilitu naproti emocionálnej stabilite,
  - nadobúda hodnoty od 0 po 1, 0 pre absenciu vlastnosti,
    1 pre maximálnu mieru vlastnosti,
  - hodnota 0 predstavuje emocionálnu stabilitu, zvláda krízové situácie, neprejavuje negatívne emócie, zostáva v každom okamihu pokojný,
  - hodnota 0,5 je rovnováhou, pri vážnych krízových situáciách reaguje podráždene, dokáže sa kontrolovať, ale dáva najavo negatívne emócie
  - hodnota 1 znamená emocionálnu nestabilitu, krízové situácie nezvláda, je známy emocionálnymi, podráždenými reakciami.

Z uvedených definícií piatich vlastností plynie, že vlastnosť O sa bude prejavovať pri ochote generovať extra úlohy pre žiakov, aby tak boli motivovaní k lepším výsledkom. Vo všeobecnosti pôjde o to, že čím lepšia a kvalitnejšia práca na riadnych úlohách, tým väčšia šanca pridelenia úlohy navyše. Vlastnosť C sa bude prejavovať v tolerancii na neskoré odovzdávanie ako aj v prideľovaní ďalších termínov na plnenie úloh. Vlastnosť E sa prejaví vo formovaní odpovede, teda emailu, ktorý študent obdrží. Generované budú stručné odpovede pre introvertné osobnosti, menej stručné pre extrovertné. Vlastnosť A sa ukáže v posudzovaní kvality, čím nižšia hodnota, tým väčšia prísnosť v hodnotení kvality projektu. Napokon, vlastnosť N sa premení do generovania upozorňovacích správ, keď sa bude približovať termín odovzdávania, teda neurotická osoba bude študentov poháňať do práce zväčšeným počtom správ.

5.2 Vytvorenie profilov členov virtuálnej firmy pre podporu výučby

Pre priblíženie sa k reálnej tvorbe softvérových produktov vo firmách, pri výučbe programovania využijeme viacero osôb, ktoré budú interagovať s jednotlivými študentami. Konkrétne budú vytvorené tri postavy:

- pán Müller - riaditeľ virtuálnej firmy,
- Tatiana - produktový manažér,
- Adam - hlavný vývojár produktu.

Osobnostné profily zamestnancov firmy sú uvedené v nasledujúcej tabuľke (Tabuľka 5 – 1).

Zamestnanec/Vlastnosť	O	C	E	A	N
pán Müller	0.0	0.2	0.2	0.5	0.0
Tatiana	0.3	0.8	0.4	0.7	0.8
Adam	0.6	0.4	0.6	0.3	0.2

Tabuľka 5 – 1 Osobnostné profily zamestnancov virtuálnej firmy

Na základe údajov z tabuľky môžeme zosumarizovať osobnostné profily zamestnancov v nasledovnom prehľade.

- Pán Müller:
  - extra úlohy nezadáva, lebo to robia jeho podriadení, teda žiadnym spôsobom nezohľadňuje originalitu,
  - termíny študentom buď neudáva a nekontroluje vôbec, alebo tomu neprikladá váhu, je to šéf, ktorý je neorganizovaný, spontánny, pracuje podľa vlastných pravidiel,

  - správy odosiela úplne stručné, má čas sa vyjadrovať len k dôležitým veciam, je nezhovorčivý introvert,
  - na kvalitu si potrpí, ale primerane, má priemerné nároky na ostatných, rovnováha medzi profesionalitou a priateľskosťou,
  - neposiela upomienky, nič ho nevytočí zo stránky študentov, na to sú jeho podriadení.

- Tatiana:
  - dáva priestor na extra úlohy, ale o polovicu menší ako Adam, je uzavretá voči originalite, zaujímajú ju hlavne tradičné postupy,
  - termíny a časový predstih sú prioritou, udáva prísne termíny, je vysoko organizovaná, zameraná na dochvíľnosť, neznesie meškania,
  - má pomerne strohé vyjadrovanie, posiela krátke správy, je miernou introvertkou,
  - nemá prehnané nároky na kvalitu, v bežných podmienkach je priateľská, chápavá, v samotnej práci nevyžaduje maximum,
  - v krízových situáciách posiela správy v jednom kuse, má zlostné pripomienky, nevyrovnaná osoba, teda krízové situácie a stres zvláda ťažko.

- Adam:
  - dáva určitý priestor na extra úlohy, oceňuje snahu, je mierne otvorený voči originalite, nevyhľadáva ju, ale vie ju oceniť,
  - oproti Tatiane kladie polovičný dôraz na termíny, voľnejšie zadávanie úloh, má zdravý pohľad na dochvíľnosť, nie je prehnane organizovaný ani zodpovedný,

– odosielané správy v porovnaní s kolegami sú najdlhšie, je mierne extrovertný,

– kvalita je preňho priorita, detailista, kritizuje aj malé chyby, je profesionálny, má vysoké nároky, malý priestor pre chyby iných,

– upomienky posiela len tesne pred termínom, je pokojný, emocionálne stabilný, neprejavuje veľmi negatívne emócie.

6 Od nápadu k produktu

„Najťažšou vecou v živote je pretavenie myšlienky
do aktivít a výsledkov."
Johann Wolfgang von Goethe

V predchádzajúcich kapitolách mal čitateľ možnosť získať predstavu o aktuálnom stave výučby predmetov o programovaní, nie len na univerzite v slovenskom prostredí, ale aj o situácii vo svete. Zadefinovaním programovania ako zručnosti nevyhnutnej pri práci softvérového inžiniera sme dospeli k záveru, že výučba programovania ochudobnená o prepojenie s procesmi, ktoré prebiehajú pri tvorbe softvérových produktov je zrelá na inováciu. Možnosť vyučovania programovania v kontexte softvérového inžinierstva sme objavili vo vytvorení virtuálnej spoločnosti, ktorá bude pristupovať k študentovi počas jeho práce na projekte ako k riadnemu členovi vývojarského tímu, bude s ním interagovať v rámci kolegiálnych vzťahov na pracovisku, aby tak študent dosiahol pocit skutočnej existencie nami vytvorenej fiktívnej spoločnosti.

Interakcia s kolegami je len jednou stranou mince v pracovnom prostredí softvérového inžiniera. Tou druhou je, prirodzene, vývoj produktu. Úlohou študenta je vytvoriť reálne použiteľný program, konkrétne hrateľnú verziu hry míny. Tento projekt je prístupný ostatným členom pracovného tímu, ide teda o kód, ktorý je zdieľaný. Práve tento fakt nám umožňuje študentovi vysvetliť prínosy softvérových produktov, ktoré sú určené pre projektovú komunikáciu. Skĺbením práce žiaka s pracovnou klímou poskytovanou jeho kolegami dosiahneme, že študent získa poznatky o výkone povolania softvérového inžiniera v praktickom živote. Po pripomenutí základných myšlienok tejto práce sa spoločne pozrime na ich realizáciu, ktorú si kvôli komplexnosti riešenia rozdelíme na niekoľko častí.

6.1 Pozícia študenta v projekte integrovania procesov SI do výučby predmetov o programovaní

Je nutné nezabúdať, že prioritnou úlohou predmetov o programovaní je naučiť študentov práve činnosť vytvárania softvérových produktov v zmysle písania zdrojových textov programu. Poslucháč má za úlohu vytvoriť pomerne rozsiahly produkt, pri vývoji ktorého si osvojí základné zručnosti pri práci s konkrétnym programovacím jazykom a vývojovým prostredím. Cieľom tejto práce je zmeniť spôsob akým sa týmto zručnostiam naučí v rámci procesov spätých s vývojom softvéru, teda v simulovanej spolupráci s fiktívnou firmou.

V úvode simulácie je potrebné študenta začleniť do tímu. V praxi to znamená, že je konkrétnemu študentovi pridelená spoluúčasť na projekte. O spoluúčasti hovoríme z dôvodu prijatia študenta do pracovného prostredia, v ktorom pracuje na projektoch firmy a nie svojich súkromných projektoch. V ďalšom napredovaní kurzu je tento projekt implementovaný do finálnej podoby, v ktorej by mal byť plnohodnotným softvérovým produktom. Aby tomu tak skutočne bolo je nutná kontrola postupu študenta, ktorá bude vykonávaná učiteľmi v školskom prostredí ako aj spolupracovníkmi z firemného prostredia. Keďže kontrola zadaní a riadenie postupu je prenesené najmä na fiktívne postavy a nie reálnych lektorov, tieto činnosti je nutné automatizovať. Automatizácia v našom prípade znamená nutnosť vykonania nasledujúcich troch krokov:

- získanie aktuálnej verzie riešenia študenta,
- vyhodnotiť správnosť aktuálnej verzie riešenia študenta,
- na základe vyhodnotenia správnosti riešenia je potrebné vypracovať stratégiu ďalšieho postupu vývoja softvéru.

6.2 Metódy a postupy získania študentských riešení

Každý poslucháč je v procese výučby konkrétneho predmetu organizovaný do určitej hierarchie. Najjednoduchšia hierarchia predstavuje model, v ktorom sa študent nachádza v jednej štúdijnej skupine. Komplikovanejšie hierarchie vytvárajú v rámci tejto skupiny podskupiny, ktoré predstavujú množinu ľudí pracujúcich ako tím. Z myšlienok tejto práce vyplýva organizáciu študenta v komplikovanejšej hierarchii, kedy sa ocitá v tíme ľudí(aj keď je zvyšok tímu fiktívny). K tejto hierarchii je potrebné prispôsobiť organizáciu softvéru projektovej komunikácie tak, aby sa v nej vedeli ľahko zorientovať všetky tri strany: študent, lektor a automatizovaný systém. Softvér projektovej komunikácie GitLab bol zvolený pre tento experiment aj z dôvodu podpory spomínaného hierarchického modelu.

Prostredie GitLab-u umožňuje vytváranie rozličných skupín, pre ktoré je charakteristické zdieľanie projektov. Tento fakt sa dá výborne využiť, veď napokon, každý študent má projekt zdieľaný so skupinou iných ľudí virtuálnej firmy. Problémom ostáva začlenenie tejto skupiny do rodičovskej skupiny, ktorá by predstavovala jednu študijnú skupinu. Súčasná verzia GitLab-u takúto hierarchiu nepodporuje, takže týmto smerom sa nie je možné uberať. Napriek zmienenému faktu, skúsme využiť pojem skupiny tak ako ho chápe nástroj projektovej komunikácie na vytvorenie štúdijnej skupiny. Tu zistíme, že študijnú skupinu môžeme efektívne vytvoriť a teda najuniverzálnejšieho člena hierarchie máme reprezentovaného GitLab-ovskou skupinou. Následne je nutné vytvoriť podskupiny, v ktorých budú tvoriť študenti. Riešenie tohto problému sa javí ako triviálne. Každý projekt, ktorý je niekým vytvorený ma možnosť byť zdieľaný s inými používateľmi, ktorým je možné priradiť i odobrať práva s akými môže s projektom manipulovať. Takýto projekt ale nesmie byť súčasťou skupiny z dôvodu viditeľnosti projektu všetkými členmi tejto skupiny a tu vzniká riziko kopírovania a pirátstva medzi členmi skupiny.

Projekty, ktoré študenti vypracuvávajú počas cvičení predmetov o programovaní sú

vopred pripravené, aby tak študenti nezačínali tak povediac "na zelenej lúke". Veď napokon aj v našom prípade sa stanú členmi tímu so začatým projektom. Z toho vyplýva, že určitá časť kódu je v každom projekte rovnaká. Nevýhoda kopírovania a pirátstva spomínaná vyššie v práci sa tu ocitá v pozícii výhody a môže byť využitá následovne. Každá študijná skupina má vytvorený jeden spoločný projekt, v ktorom sa počas semestra budú objavovať zdrojové texty programov, ktoré študenti neimplementujú ale využívajú a dopĺňajú. Pozornému čitateľovi neunikne fakt, že to isté sa dá zrealizovať aj na projektoch mimo skupiny, avšak prišli by sme o niekoľko výhody vyplývajúcich z využitého prístupu:

- študent sa naučí pracovať s viacerými projektami súčasne, čo sa v praxi bežne stáva,
- spoločný projekt je viditeľný všetkými členmi tímu, čo implikuje zdieľanie diskusného fóra daného projektu.

Diskusné fórum projektu umožní študentom, lektorom aj automatizovanému systému priestor na interakciu pri rôznych problémoch, ktoré sa vyskytnú v procese riešenia softvérového projektu. Zúčastnené strany si môžu vymieňať informácie od návodov pre nastavenia vývojarského prostredia až po pomoc pri riešení implementačných záležitostí.

Uvedené začlenenie študenta do tejto štruktúry hierarchie je osožné najmä pre lektora. Vyplýva to z vlastnosti systému, že nájsť študentovo zadanie je možné aj iteratívnym prehľadávaním zadaní všetkých študentov, ale to by malo vplyv na rýchlosť a optimálnosť systému. Skupinové členenie tiež umožňuje ďalší vývoj tejto práce pre potreby generovania analýz údajov získavaných v priebehu semestra, na základe ktorej bude mať lektor relevantné informácie o stave svojich študentov a študenti si budú vedieť navzájom porovnávať dosiahnuté výsledky svojich riešení.

Každý študent pracuje na projekte, ktorý priebežne odovzdáva do systému pre

správu verzií do projektu zdieľaného s členmi virtuálnej firmy, ktorí tieto projekty vytvárajú. Podobne, jednotlivé študentské projekty majú rovnakú štruktúru. Z týchto dvoch skutočností vyplýva, že je možné automaticky zostaviť odkaz na stránku, kde sa nachádza konkrétna časť zdrojového textu programu študenta. Takýto odkaz môže vyzerať následovne:

https://git.cnl.sk/pp318cw/ld851qu/.../Minesweeper.java

Tento odkaz poukazuje na miesto, kde sa nachádza zdrojový kód triedy *Minesweer*, ktorá sa nachádza v projekte pridelenému študentovi s identifikátorom *ld851qu* a projekt vytvoril používateľ s identifikátorom *pp318cw*. Je možné takýmto spôsobom prejsť všetkými zdrojovými textami projektu iteratívnym spôsobom s aktuálnou adresou a postup opakovať pre všetkých študentov, ktorých identifikátory si vieme získať z príslušnosti študenta ku skupine, ktorej lektorom je napríklad človek s identifikátorom *pp318cw*. Pre zhrnutie môžeme konštatovať, že princíp získania študentského zadania pozostáva z týchto krokov:

- zistenie identifikátora lektora skupiny,
- na základe identifikátora lektora skupiny získať identifikátory členov skupín (študentov), ktorí do danej skupiny patria,
- pre každého člena skupiny vygenerovať odkazy na stránky so zdrojovými textami riešenia na základe znalosti štruktúry riešenia.

Po zvládnutí získania projektu, na ktorom pracuje poslucháč môžeme pristúpiť k fáze vyhodnoteniu projektu.

6.3 Metódy a postupy vyhodnotenia študentských riešení

Pomocou známej štruktúry projektu si vytvoríme kópiu projektu na serveri, ktorý spúšťa aj tento automatizovaný systém riadenia projektov pre študentov. Práve do tohto projektu sa sťahujú zdrojové texty programov študentov z odkazov, generovanie ktorých sme si už v tejto práci objasnili. Pripravený študentský projekt je následne spustený našim produktom, ktorý čaká na jeho vykonanie. Študentský projekt obsahuje knižnicu, ktorá umožní jeho syntaktickú aj sémantickú kontrolu. Táto knižnica bola úspešne implementovaná a otestovaná v práci Denisy Šebejovej (Šebejová, 2012). Princípy autorkinej práce pre korektnosť priblížim aj v tejto práci.

Štrukturálna kontrola je navrhnutá a implementovala tak, aby viedla študentov k správnemu riešeniu zadania podľa princípov objektového programovania. To znamená, že sa kontrolujú modifikátory prístupu premenných, metód, tried a podobne. Rovnako, študenti si pozmeňujú predpripravený zdrojový text programu tak, aby vyhovoval ich riešeniu a nerešpektujú tak zásady objektovo-orientovaného programovania. Táto kontrola teda otestuje štruktúru programu, kedy ale nejde o kontrolu postupu študenta, či ho niečo brzdí vo vývoji alebo podobne. Hlavnou úlohou kontroly je držať ochrannú ruku nad dodržiavaním princípov OOP v študentovom riešení. Nástroj funguje na princípe použitia nepriamej reprezentácie správneho riešenia, ktorého jadro tvorí anotačný procesor. Systém porovnáva elementy študentského a správneho riešenia.

Sémantická kontrola slúži na otestovanie požadovanej funkcionality riešenia študenta. Testovanie je založené na princípe testovania časti aplikácie, kedy sa tento fragment kódu izoluje od ostatných a testuje sa ako samostatná jednotka. Nevýhodou ostáva odizolovanie sa od zvyšku aplikácie. Riešenie je koncipované tak, že kontrola je spúšťaná automaticky po vytvorení súboru JAR zo študentského zadania. Celé študentské riešenie sa teda spustí ako samostatný program, ktorý vytvorí súbor pod názvom *builds.xml*. Súbor obsahuje nedostatky, ktoré sa vyskytli pri spustení

projektu študenta. Tieto nedostatky môžu byť chybou štruktúry programu alebo jeho sémantiky. Príklad súboru je možné vidieť na obrázku(Obr. 6 – 1).

```
<?xml version="1.0" encoding="windows-1250" standalone="no"?>
<failures>
    <module number="02">
        <test_class name="tests.module2.FieldTest">
            <class_name>minesweeper.core.Field</class_name>
            <method_name>getRowCount()</method_name>
            <task_id_look>(pozri http://hornad.fei.tuke.sk/~poruban/java/04/index.html#getters)</task_id_look>
            <task_id>getters</task_id>
            <reason>nesprávne vytvorený getter;  expected:&lt;10&gt; but was:&lt;7&gt;</reason>
            <line>/Users/frantisekklucivjansky/NetBeansProjects/MinesweeperTest/src/minesweeper/Field.java:180</line>
        </test_class>
    </module>
</failures>
```

Obr. 6 – 1 Ukážka obsahu súboru builds.xml

Z obrázka(Obr. 6 – 1) je zrejmé, že vieme jednoznačne identifikovať názov triedy, v ktorej sa objavila problémová časť kódu, identifikátor úlohy, ktorú sme mali vypracovať a dokonca máme k dispozícii aj konkrétny riadok výskytu chyby. Celý súbor *builds.xml* je tak vstupom do ďalšej časti nášho automatizovaného systému pre riadenie postupu študentov v zmysle dodržiavania pravidiel a postupov softvérového inžinierstva.

6.4 Metódy a postupy vypracovania stratégie ďalšieho postupu vývoja softvéru

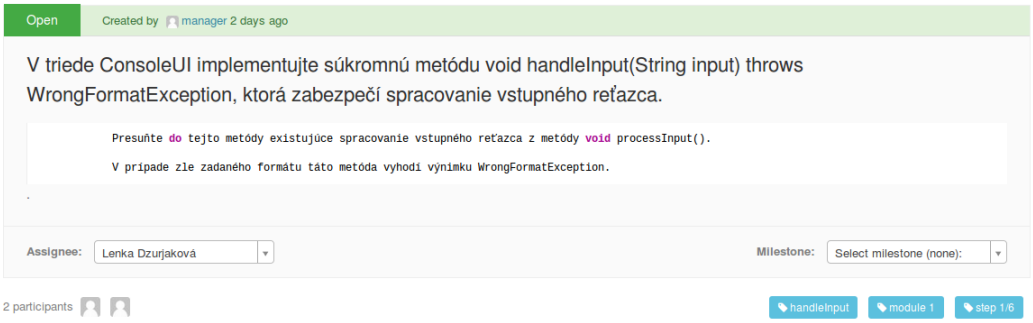
Poslednou fázou prítomnou v procese automatizácie riadenia postupu študentov je rozhodnutie o ďalších krokoch. V princípe môžu nastať dve situácie.

- Študent svoje zadanie splnil podľa všetkých pravidiel a sú mu vygenerované ďalšie úlohy na implementáciu.
- Študent svoje zadanie nesplnil v požadovanej kvalite a musí úlohu prerobiť.

V obidvoch prípadoch sa spomínajú úlohy, ktoré má študent splniť. Pozrime sa teda bližšie na techniku vytvárania úloh pre študenta. Systém GitLab umožňuje ku každému projektu vytvárať množinu úloh, ktoré je potrebné na projekte vykonať. Táto

úloha je následne pridelená niektorému členovi z projektového tímu. Systém taktiež umožňuje pridávať k úlohám rozličné označenia, akými sú *dôležité*, *urgentné* a pod. Vráťme sa naspäť k našej práci. Je potrebné kontrolovať správnosť študentského riešenia a to na základe toho, či splnil niektorú úlohu. Ako ale zistíme, na ktorej úlohe študent skončil svoje riešenie? Riešenie sa naskytá v podobe vytvorenia dvoch rovnakých úloh. Prvá je určená a pridelená študentovi, druhá niektorému zo zostávajúcich členov. Z obmedzení, ktoré vyplývajú z role študenta na projekte vyplýva, že študent môže manipulovať len s úlohami, ktoré boli pridelené jemu. Po vypracovaní úlohy študent označí svoju úlohu ako ukončenú. Ukončené úlohy sa porovnajú s tými, ktoré sú neukončené a patria inému členovi tímu. Pri zhode identifikátorov úloh sa tieto úlohy prefiltrujú a len tieto sú podrobené testovaniu spomínanému vyššie v práci. Je dôležité, aby práve študent označil svoje úlohy ako ukončené, keďže aj v praxi sa môžu kontrolovať len tie úlohy, ktoré ich riešiteľ považuje za hotové, prípadne rozpracované. Pravdou ale ostáva, že systém GitLab pozná len dva rôzne stavy úloh, konkrétne otvorená úloha a úloha ukončená.

Úlohy sa študentom vytvárajú postupne tak, ako plynie semester. Každá úloha je priradená k určitému kroku v rámci modulu. Trvanie modulu zodpovedá dĺžke jedného týždňa. Na nasledujúcom obrázku(Obr. 6 – 2) je možné vidieť úlohu pre študenta, ktorá obsahuje jednotlivé informácie o moduloch a krokoch. Po úspešnom vypra-



Obr. 6 – 2 Ukážka úlohy v systéme GitLab

covaní úloh z niektorého kroku sa študentom vygenerujú nové úlohy, ktoré sú buď

z ďalšieho kroku rovnakého modulu alebo z prvého kroku nového modulu. Samotné generovanie nových úloh by ale nebolo úplne ekvivalentné k procesom, ktoré prebiehajú vo firmách. Vo firemnom priestore totiž dochádza k sociálnej interakcii, kedy členovia tímu rozprávajú o práci medzi sebou. Môže ísť o pochvaly za dobre vykonanú prácu, na strane druhej to môžu byť pokarhania za výskyt problémov. Ako sa s týmito záležitosťami vyrovnáva náš systém je predmetom ďalšej časti tejto práce.

6.5 Sociálna interakcia v systéme riadenia postupu študenta

Vychádzajúc z tézy tejto práce pojednávajúcej o priblížení procesov, ktoré prebiehajú pri vytváraní softvéru, je nevyhnutné začleniť študenta do tímu kolegov a zabezpečiť, aby táto skupina navzájom komunikovala. Projektovú komunikáciu na báze prideľovania nových úloh do prostredia GitLabu sme si už predstavili a rovnako boli spomenuté aj isté obmedzenia týkajúce sa tohto prístupu. Úloha vytvorená pre študenta ostáva len úlohou, ktorú je potrebné vyriešiť, teda vo firemnom prostredí je podstatný výsledok vypracovanej úlohy. Nesmieme ale zabúdať, že každá úloha sa rieši v určitom pracovnom postupe a počas tejto práce dochádza ku kontaktu s kolegami. Uvedená interakcia medzi spolupracovníkmi sa môže líšiť od debát o správnosti prístupu k riešeniu, o odhaľovaní chýb v kóde, o zisťovaní stavu vypracovania úlohy v prípade, že spolupracovník je odkázaný na výsledky vypracovanej úlohy a podobne.

Pri integrácií sociálnej interakcie do celého systému vytvoreného v rámci tejto práce je dôležité nezabúdať, že jedinou reálnou súčasťou tímu je študent, čo implikuje nemožnosť priamej komunikácie. Vzájomná korešpondencia je teda vyslovene písomná, pričom sa dôraz kladie na dôveryhodnosť obsahu správ, aby pôsobili dojmom vytvorenia skutočnými osobami. Vytvorenie dojmu autenticity je kľúčové pre potreby tejto práce a nie je preto náhodné, že sa ctený čitateľ tejto publikácie stretol s kapitolou o modeloch ľudského charakteru z pohľadu psychológie. Predstavený mo-

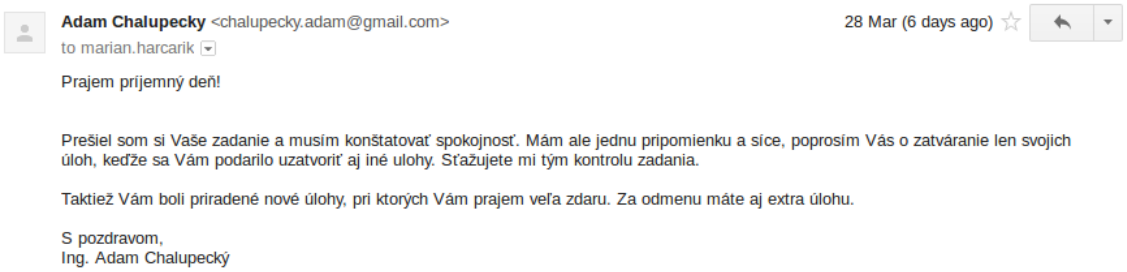
del *OCEAN* kvantitatívne popisuje jednotlivé povahové črty osobnosti, čo vytvára možnosti pre transformovanie každej z týchto čŕt do podoby matematických funkcií. Predpis jednej takejto funkcie môže byť nasledovný:

function: FLOAT x BOOLEAN → STRING

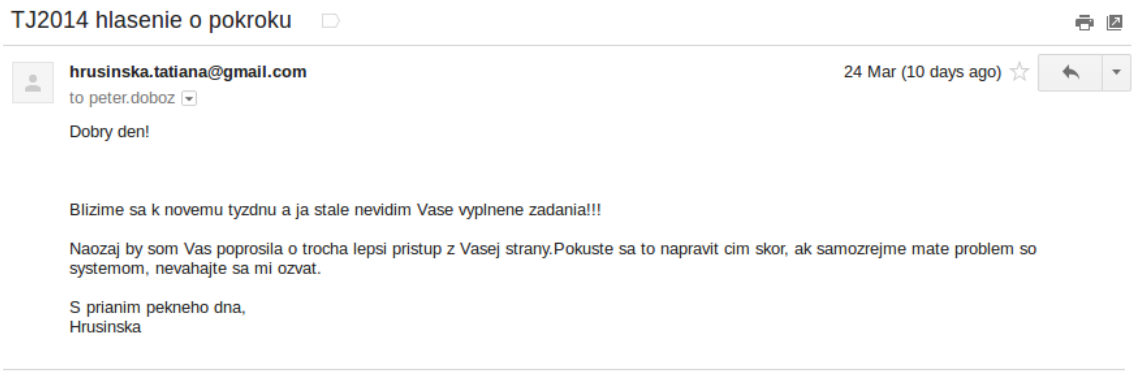
Takáto funkcia je funkciou dvoch argumentov. Prvý argument predstavuje číselnú hodnotu v rozmedzí 0 až 1 indikujúcu prítomnosť danej vlastnosti u konkrétneho jedinca (Tabuľka 5 – 1). Druhým argumentom je pre zjednodušenie pravdivostná hodnota, ktorá nesie informáciu o splnení danej úlohy, ktorá bola študentovi pridelená. Oborom hodnôt tejto funkcie je reťazec znakov, teda adekvátna odpoveď virtuálneho člena tímu.

Vytváranie správy prebieha nasledovným spôsobom. V prípade, že sa študentovi vyhodnotí zadanie ako správne vyriešené, vygeneruje sa telo správy, ktoré nesie pochvalný obsah. V závislosti na povahových vlastnostiach osoby(konkrétne na vlastnosti extrovercie) je obsah správy buď stručný alebo rozsiahlejší. V prípade, že je úloha nesprávna, telo správy nesie informácie o chybách(štrukturálnych aj sémantických), ktoré získava ako výstup z modulu pre kontrolu zadania. Študent tak získa predstavu, čo bolo zlé a má šancu chybu napraviť. Výpis chýb je doplnený o text, ktorý má študenta motivovať k napredovaniu aj napriek tomu, že urobil zopár chýb. Príklady komunikácie sú uvedené na nasledujúcich obrázkoch(Obr. 6 – 3 a Obr. 6 – 4)

Uvedené ukážky obsahujú ako pochvalné vety, tak aj vety, ktoré majú študenta motivovať k usilovnejšej práci. V ukážkach sa odráža aj osobnosť postavy, ktorá email odoslala. Ako vidíme, pani Hrušinská, ktorá je náchylná na dodržiavanie termínov je nespokojná s prístupom študenta k jeho povinnostiam. Na druhej strane, výrečný pán Chalupecký je vo svojej pochvale rozsiahly a motivuje študenta v ďalšom napredovaní projektu.



Obr. 6 – 3 Ukážka pochvalnej korešpondencie



Obr. 6 – 4 Ukážka korešpondencie pri zlyhávaní študenta

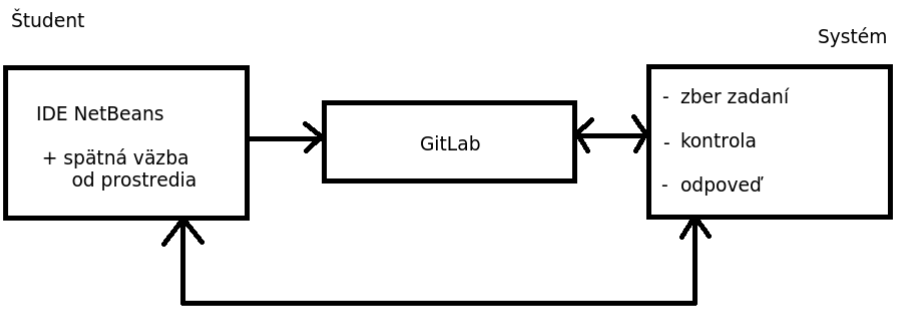
Po priblížení detailov implementácie od získavania študentských zadaní, ich kontroly a vyhodnotenia až po sociálnu komunikáciu v tíme softvérových inžinierov sa naskytá priestor na reálne nasadenie navrhnutého systému do výučby. Spoločne sa pozrieme na to, akým spôsobom sa systém zhostil svojej úlohy, či študentom naozaj prináša želané účinky, je autonómny a spoľahlivý. To všetko je v réžii nasledujúcej kapitoly. Pre prípadné ďalšie detaily implementácie sa odporúča nahliadnuť do systémovej príručky, ktorá je prílohou(Príloha A) tejto publikácie.

7 Od produktu k nápadom

„Úloha človeka - vedca nespočíva v zameraní sa na okamžité výsledky.
Rovnako neočakáva, že jeho pokrokové myšlienky budú hneď prijaté.
Jeho úlohou je pripraviť pôdu pre tých, ktorí ho budú následovať
a ukázať im cestu."
Nikola Tesla

Celé snaženie predkladanej práce vedie k zlepšeniu výučby predmetov o programovaní v zmysle zavedenia procesov, ktoré prebiehajú pri tvorbe softvéru do študijných materiálov a osnov spomínaných predmetov. Študent sa ocitá v centre záujmu práce a je nevyhnutné, aby sa jeho vyjadrenia o používaní navrhnutého riešenia objavili aj v tejto publikácii. Pred samotnými výpoveďami študentov je dôležité načrtnúť čitateľovi podmienky, v ktorých poslucháči pracovali, aby bolo možné lepšie pochopiť dosiahnuté hodnotenie systému.

Prostredie systému, v ktorom účastníci testovania pracovali ilustruje nasledujúci obrázok (Obr. 7 – 1). Z obrázka(Obr. 7 – 1) vyplýva, že študent pracuje lokálne na svo-



Obr. 7 – 1 Ukážka prostredia, v ktorom pracuje študent

jom(prípadne školskom počítači) vo vývojovom prostredí NetBeans. Toto prostredie

bolo zvolené kvôli už existujúcej nástrojovej podpore sémantickej a štrukturálnej kontroly tak, ako to bolo opísané v tejto práci. Následne úlohou študenta je vypracovávané zadanie posielať do systému pre správu verzií projektov, v našom prípade ide o systém GitLab.

Každý študent mal v prostredí pre správu verzií projektov vytvorený svoj samostatný projekt, pričom tento projekt mal byť zdieľaný so zástupcami firmy. Tu sme sa po prvýkrát odčlenili od plánov zamýšľaných v analytickej časti práce, keďže projekt bol zdieľaný len s autorom tejto práce a nie so zamýšľanými osobami - pani Hrušinskou a pánom Chalupeckým. Odôvodnenie tohto kroku je podmienené skutočnosťou, že s testovaním systému sa začalo až počas prebiehajúceho semestra a z časových dôvodov sa nevytvorili v systéme ďalšie kontá pre spomínaných ľudí. Každopádne, členovia virtuálneho tímu neboli so systému vyradení, ich úloha je v projekte naďalej podstatná.

Projekt študenta je teda zdieľaný len s jednou osobou, avšak generovanie nových úloh je ponechané na systém, pričom sa využíva všeobecný administrátorský účet vytvorený v tomto prostredí, konkrétne je to účet manažéra. Je teda ponechaný dojem zásahu do projektu treťou stranou(virtuálnou firmou) a nie len interakcia študent a autor tejto práce. Zachovaná ostala aj myšlienka generovania nových úloh pre oboch členov projektu, kedy sa študentovi priradila úloha na vypracovanie a rovnakú úlohu dostal druhý člen tímu pre kontrolu postupu študenta. Princíp duplicitného vytvárania úloh je nevyhnutný pre správnu prácu kontrolóra zadania.

Ďalšou podstatnou časťou pracovného prostredia študenta je účasť na spoločnom projekte, ktorý je zdieľaný medzi všetkými členmi testovanej vzorky študentov. V tomto projekte sú k dispozícii nápovede pre prácu na zadaní, informácie o správnom nastavení prostredia a podobne. V spoločnom prostredí sa postupne objavujú aj zdrojové kódy projektu, ktoré je nutné študentom odovzdať kvôli ďalšiemu postupu na projekte. K dispozícii je aj diskusné fórum, ktoré je možné využívať pri rozličných

problémoch, s ktorými sa študent v priebehu semestra potýka. Pracovné prostredie z pohľadu študenta máme predstavené, nasleduje pohľad automatizovaného systému.

Systém automatického riadenia procesov prebiehajúcich pri tvorbe softvéru má, podobne ako študent, prístup ku všetkým zadaniam, na ktorých je prítomný aj poslucháč. Kvôli zvýšeniu pocitu skutočnej firmy je systém spúšťaný v náhodných časoch a dňoch, keďže súčasťou systému je aj sledovanie postupu študenta a ten sa nemusí sledovať na dennej báze. Pri spustení systému dôjde k následnému stiahnutiu študentských zadaní ich vyhodnoteniu a sociálnej interakcii, ktorá pozostáva z dvoch častí. Najprv sa vygeneruje nový pár z každej úlohy pre študenta(respektíve sa obnoví predošlá úloha) a odošle sa správa vo forme elektronickej pošty. Pre odosielanie pošty boli použité účty členov virtuálnej firmy, ktorí takýmto spôsobom ostali v systéme tak, ako to predpokladala analytická časť riešenia tejto práce. Cez zadefinované pracovné prostredie sa následne môžeme preniesť do samotného predstavenia testovaného tímu.

7.1 Študeni v úlohe účastníkov projektu automatizovanej kontroly postupu študentských riešení

Pre potreby testovania prínosov tejto práce bola náhodne zvolená päťčlenná skupina študentov, ktorým patrí osobitá vďaka autora tejto práce a ktorých mená sú uvedené v ďakovnej reči riešiteľa práce. Títo študenti kurzu "Technológie JAVA"boli vybraní v piatom týždni semestra, kedy výučba kurzu prebiehala v plnom prúde a teda podľa predchádzajúcej formy výučby. Oneskorenie integrácie navrhovaného riešenia do procesu výučby bolo spôsobené technickými problémami v implementačnej fáze vývoja softvéru. Oneskorenie malo veľký vplyv na celé testovanie, keďže sa testovacej vzorke študentov nepodaril navodiť pocit reálnej existencie spomínanej firmy. Už pri výbere študentov sa testovanie systému predstavilo ako študentské zadanie, ktoré môže pomôcť ich nasledovníkom, ktorí tento kurz absolvujú v ďalších rokoch.

Ďalšou nevýhodou, ktorá je vnímaná z pohľadu autora práce je prístup študentom k materiálom pôvodnej formy organizácie cvičení. Študenti tak mali prehľad o tom, čo majú implementovať a úlohy často krát mali vyriešené skôr než im boli systémom pridelené v prostredí GitLab-u. Tieto úlohy tak následne mali vyriešené správne a len sporadicky sa stávalo, že systém vygeneroval správu obsahujúcu nedostatky riešenia. Generované teda boli poväčšine pochvalné správy pre študentov a nebolo možné v celej miere otestovať generovanie negatívnych správ. Spomínané postrehy autora práce sú len jednou stranou mince a rovnako podstatné sú aj postrehy študentov. Ich hodnotenie, kritika i námety na zlepšenie boli získané formou dotazníka, ktorého vyhodnotenie je v nasledujúcej tabuľke (Tabuľka 7 – 1). Legenda k tabuľke je v tabuľke 7 – 2.

Úlohou dotazníka bolo získať základné informácie priamo od študentov. Prvé tri otázky sú zamerané na pracovné skúsenosti žiakov. Tieto údaje sú zaujímavé najmä z toho hľadiska, že ak študent má pracovné návyky z oblasti informačných technológií, tak poskytne veľmi dôležitý pohľad na nami vytvorené riešenie. Bude vedieť porovnať našu projektovú komunikáciu s komunikáciou vo svojej spoločnosti. Z prieskumu vidno, že pracovné skúsenosti nemá ani jeden korešpondent. To je pozitívne z pohľadu prínosu práce pre študentov, ktorí môžu nadobudnuté poznatky z tohto kurzu využiť v budúcej praxi. Je tiež pozitívne, že väčšina respondentov sa už stretla s nástrojmi projektovej komunikácie na predošlých predmetoch. Je to signál, že študenti získavajú informácie o projektovej komunikácii už na predošlých kurzoch.

Ďalšia sada otázok bola zameraná na skúsenosti s programovacím jazykom JAVA. Tento prieskum je podstatný z dôvodu prínosu navrhovanej formy výučby pre študentov, ktorí už v danom jazyku programovali a rovnako aj pre tých, ktorí sú v tomto jazyku úplní začiatočníci. Z prieskumu vyplynulo, že väčšina študentov už skúsenosti s jazykom JAVA má, opäť tieto vedomosti boli nadobudnuté v predošlých kurzoch na našej univerzite.

Odpovede študentov					
Otázka/respondent	1	2	3	4	5
1.Máte skúsenosti s prácou v IT firmách?	nie	nie	nie	nie	nie
2.Prišli ste v minulosti do kontaktu s nástrojmi projektovej komunikácie(NPK)?	áno	áno	áno	áno	áno
3.S akými NPK ste prišli do kontaktu?	git	git	git,SVN	git	git
4.Máte predchádzajúce skúsenosti s programovacím jazykom JAVA?	áno	áno	áno	áno	áno
5.Ak máte predchádzajúce znalosti z jazyka JAVA, kde ste ich nadobudli?	1	1	1	1,2	1,2
6.Pri výučbe technológií JAVA Vám forma štúdijných materiálov:	2	1	2	2	2
7.Ak ste nespokojní s formou cvičení, ako by ste ju navrhovali zmeniť?	x	x	x	x	x
8.Páčila sa Vám myšlienka prepojenia výučby programovania s procesmi, ktoré prebiehajú pri tvorbe softvéru?	áno	áno	áno	áno	áno
9.Páčilo sa Vám testované riešenie integrácie procesov tvorby softvéru s výučbou softvéru?	áno	áno	áno	áno	áno
10.Boli úlohy v systéme GitLab formulované dostatočne jasne?	áno	nie	nie	áno	nie
11.Aké nedostatky ste vnímali pri testovaní navrhovaného riešenia?	1	2	1	1	2
12.Vedeli by ste si predstaviť takúto formu cvičení na predmetoch o programovaní?	1	1	1	1	1

Tabuľka 7 – 1 Odpovede študentov z dotazníka o spokojnosti s testovaným systémom

hodnota/otázka	5	6	11	12
1	škola	vyhovuje úplne	žiadne	áno s ponechaním úloh
2	online	skôr vyhovuje	úlohy	x

Tabuľka 7 – 2 Legenda k odpovediam študentov

Podstatná informácia sa týka pohľadu študentov na aktuálnu formu cvičení. Je potešujúce, že väčšina korešpondentov nie je spokojná s momentálnym stavom, pretože to implikuje potrebu venovať sa zmene študijných materiálov, čomu sa venuje aj táto práca. Rovnako užitočnou pripomienkou je aj inklinovanie študentov k myšlienke prepojenia procesov prebiehajúcich pri tvorbe softvéru s vyučovaním programovania. Pohľad na programovanie v širšom kontexte, teda v kontexte, pri ktorom implementácia predstavuje len jednu fázu vývoja softvéru je nevyhnutný pre profesiu softvérového inžiniera a je dobrým znamením, že to vnímajú aj študenti.

Záverečné otázky prieskumu sú venované prínosom tejto práce a odpovede nastavujú zrkadlo pre autora. Z výsledkov jasne vyplýva, že testovanie prínosu tejto práce sa vyplatilo, keďže študenti poukázali jasným spôsobom na výhody aj nedostatky práce. Z prieskumu je vidno, že najväčším nedostatkom je formulácia úloh a ich nejasné definovanie v prostredí GitLab a študentom sa lepšie pracovalo, keď mali všetky úlohy k dispozícii pohromade. Za výhody považovali kontrolovanie zadaní virtuálnymi osobami a ich kritiku aj pochvalu. Taktiež boli spokojní s prácou v systéme správy verzií projektov. Z týchto poznatkov teda môžeme konštatovať následovné. Študenti zmenu uvítali, páčilo sa im pracovať v kolektíve zamestnancov virtuálnej firmy, na druhej strane mali problémy s pochopením úloh na vypracovanie. Ďalšia práca musí zohľadniť vyjadrené stanovisko študentov, napriek tomu, že je to pomerne malá vzorka študentov. V ďalšej časti sa teda pozrime na to, akým spôsobom sa dá spomínaný nedostatok napraviť.

7.2 Študeni v úlohe bádateľov nových prístupov k zlepšeniu výučby

Vychádzajúc z výsledkov prieskumu medzi študentami (Tabuľka 7 – 1) sa v priebehu písania týchto riadkov koná ďalší experiment, v ktorom študenti vytvárajú rozšírenie svojho softvéru tak, aby bol použiteľný aj pre mobilné zariadenia. Pre tento krok sme sa rozhodli po spoločnej konzultácii so študentami, ktorí sú ochotní nadobúdať nové vedomosti nad rámec kurzu, za čo im patrí veľká vďaka. Ani jeden zo študentov, ktorí sa k tomuto kroku odhodlali, nemá predchádzajúce skúsenosti s tvorbou aplikácii pre platformu Android. Strácajú sa tak nedostatky, s ktorými sme sa museli vysporiadať pri implementovaní pôvodného projektu. Študenti tak nevedia, čo ich tentokrát čaká, nemajú vopred zadané žiadne úlohy, ktoré by mali vykonávať a komunikácia sa musí preniesť len na plecia systému GitLab. Študenti po tomto kroku môžu reálne zhodnotiť situáciu aká je bežná z praxe, kedy vopred nevedia, akým smerom sa projekt vyvinie.

Celý proces, ktorý mal byť pôvodne vyskúšaný v rámci doterajšieho scenára sa zopakuje ešte raz, keďže študenti budú pracovať v novom prostredí pre vývoj aplikácii(IDE Eclipse) a budú im prideľované nové úlohy, tentokrát kontrolované len autorom práce, keďže z časových dôvodov sa nemohli napísať testy pre kontrolu splnenia nových úloh. Plánom je vytvoriť plne funkčnú hru pre mobilné zariadenia, ktoré fungujú na platforme Android. Študenti si tak osvoja ďalšie využitie programovacieho jazyka JAVA, tvorbu aplikácií v novom vývojovom prostredí, ako aj špecifiká tvorby aplikácií pre mobilné zariadenia. Medzi tieto charakteristiky patrí využitie dotykovej plochy obrazovky ako príjemcu vstupných informácií pre aplikácie, ako aj dynamická zmena veľkosti objektov v aplikácii v závislosti od veľkosti zariadenia. Kvôli náročnosti tohto projektu budú študenti pracovať na projekte spoločnými silami, kedy si v praxi vyskúšajú ako od ich výsledkov a implementačných úspechov závisí práca ostatných členov tímu. Každý člen tak dostane za úlohu naprogramo-

vať určitú funkcionalitu a zvýši sa ich potreba vzájomnej projektovej komunikácie v porovnaní s potrebou komunikácie pri vytváraní konzolovej a desktopovej verzie hry. V prípade úspechu tohto experimentu sa naskytá možnosť integrovať implementáciu hry pre mobilné zariadenia ako doplňujúcej úlohy pre šikovných študentov v nasledujúcich rokoch, ktorí budú takýmto spôsobom odmenení hlbšími vedomosťami. Výsledky spomínaného experimentu budú zverejnené v ďalších prácach autora venovaných problematike zlepšovania kvality výučby predmetov o programovaní. K preštudovaniu spomínaných prác je ctený čitateľ tejto publikácie už teraz srdečne pozvaný a je mu autorom vyjadrená veľká vďaka, že dospel až k týmto posledným riadkom snaženia sa diplomanta.

8 Záver

V poslednej časti tejto publikácie sa nehodlám ukrývať za citáty múdrych ľudí, ktorých myšlienkami som uvádzal predošlé kapitoly, aby tak nad jednotlivými staťami držali pomyselný patronát. Nerobím to z neúcty k autoritám, ale z presvedčenia, že záver práce má formulovať výstupy práce so všetkými jej kladmi aj zápormi. A za tieto výsledky nenesie zodpovednosť nik iný ako autor, teda ja. V krátkosti sa tak pokúsim zosumarizovať vyše jednoročné snaženie sa.

Úloha implementovať systém, ktorý by automatizoval riadenie simulácie procesov softvérového inžinierstva, ktoré prebiehajú v bežnej praxi bola úspešne vyriešená v podobe funkčnej aplikácie. Táto aplikácia umožňujúca zber študentských zadaní zo systému pre správu projektov, spustenie kontroly zadania a vyhodnotenie výsledkov kontroly umožňuje efektívne simulovať reálny priebeh vývoja softvérového produktu v tíme programátorov. Vytvorená aplikácia bola otestovaná študentami, čím sa preukázala jej korektná činnosť. Fungujúci program ale nezaručuje, že prináša želaný efekt, ktorý bol cieľom práce.

Veľmi dôležité bolo nasadenie navrhovaného riešenia do vyučovacieho procesu, ktoré poukázalo na nedostatky ale aj klady aplikácie i celej myšlienky zmeny formy cvičení kurzu technológií JAVA. Je potešujúce, že študenti ponúkanú zmenu prijali, pozdáva-la sa im myšlienka prepojenia procesov vývoja softvéru a učenia sa programovania. Vo fáze testovania sa ukázala nedostatočná úprava aktuálnej verzie úloh do vhodnej podoby v systéme pre správu projektov, ktorú bude potrebné v ďalšom vývoji aplikácie pozmeniť a vylepšiť. Študenti považovali spätnú väzbu od systému ako vhodnú formu sledovania ich pokroku počas implementácie svojich zadaní, rovnako dobre prijali aj samotnú prácu so systémom projektovej komunikácie. Napriek malej vzorke testujúcich študentov možno podotknúť, že myšlienka diplomovej práce je myšlienkou hodnou rozvíjania, realizovaná podoba myšlienky je akceptovaná študentami obsahujúca nedostatky, ktoré je potrebné odstrániť v ďalšom bádaní.

Literatúra

IEEE. *„IEEE Standard Glossary of Software Engineering Terminology"* IEEE Std. 610.12-1990. Institute of Electrical and Electronics Engineers, 1990.

Naur, P. & Randell, B. *„Software En- gineering: Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968."* Scientific Affairs Division, NATO, 1969.

Hislop, Gregory W. *„Software Engineering Education: Past, Present, and Future."* Software Engineering: Effective Teaching and Learning Approaches and Practices. IGI Global, 2009.

ACM & IEEE (2013). *„Computing Curriculum 2013: The Overview Report."* IEEE Computer Society and Association for Computing Machinery.Piscataway, NJ: IEEE CS Press

McConnell, S. 2004. *Professional Software Development: Shorter Schedules, Better Projects, Superior Products, Enhanced Careers* . Boston, MA: Addison-Wesley, 2004, ISBN: 0321193679

Lethbridge, T. C., Diaz-Herrera, J., LeBlanc, R. J., Thompson, J. B. *Improving software practice through education: Challenges and future trends.* Future of Software Engineering. International Conference on Software Engineering. pp. 12-28. Piscataway, NJ: IEEE CS Press, 2007.

Robins A., Rountree J., Rountree N. 2003. *Learning and teaching programming: A review and discussion* . Computer Science Education, Vol. 13, No. 2. (June 2003), pp. 137-172.

Port D., Rachal C., Jia Liu. 2013. *Motivating and orienting novice students to value introductory software engineering.*

Nurkkala, T., & Brandle, S. 2011. *Software studio: Teaching professional software engineering.* In Pro-ceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11), 153-158.

Teel, S., Schweitzer, D. and Fulton, S. 2012. *Teaching Undergraduate Software Engineering Using Open Source Development Tools.* In Issues in Informing Science and Information Technology, Volume 9, 2012.

Shihong, H., Distante, D. 2006. *On Practice-Oriented Software Engineering Education.* In Proceedings of the 19th Conference on Software Engineering Education and Training Workshops (CSEETW'06). 2006.

Gnatz, M., Kof, L., Prilmeier, F., Seifert, T. 2003. *A Practical Approach of Teaching Software Engineering.* In Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03). 2003.

SCRUM Methodology. dostupné na internete `http://scrummethodology.com/` pristúpené 2-November-2013

VersionOne. State of agile survey: The state of agile development. dostupné na internete `http://bit.ly/12NVx6U`, 2012. pristúpené 2-November-2013.

Scharf, A., Koch, A. 2013. *Scrum in a Software Engineering Course: An In-Depth Praxis Report.* In 26th International Conference on Software Engineering Education and Training. San Francisco, CA, USA. 2013.

Tamburri, D.A., Razavian, M, Lago, P. 2013. *Teaching Software Design with Social Engagement.* In 26th International Conference on Software Engineering Education and Training. San Francisco, CA, USA. 2013.

Wiegers, K. E. 2002. *Peer Reviews in Software: A Practical Guide.* Reading, MA: Addison-Wesley, 2002.

Garousi, V. 2010. *Applying Peer Reviews in Software Engineering Education: An Experiment and Lessons Learned.* In IEEE Transactions on education, VOL. 53, NO. 2, MAY 2010.

Herold, M.J., Lynch, T.D., Ramanathan, J.,Ramnath, R. 2010. *Student and Instructor Experiences in the Inverted Classroom.* In 26th International Conference on Software Engineering Education and Training. 2012.

Qing Zhu, Tao Wang, and Shenglong Tan. 2007. *Adapting Game Technology to Support Software Engineering Process Teaching: From SimSE to MO-SEProcess.* In Third International Conference on Natural Computation. 2007.

Sureka, A., Gupta, M., Sarkar, D. a Chaudhary, V. 2013. *A Case-Study on Teaching Undergraduate-Level Software Engineering Course Using Inverted-Classroom, Large-Group, Real-Client and Studio-Based Instruction Model.* arXiv:1309.0714, 2013

Jianmin Zhang, Jian Li. 2010. *Teaching Software Engineering Using Case Study.* Proceedings of International Conference on Biomedical Engineering and Computer Science, pp. 1-4,Wuhan, Apr. 2010.

Slovenská komora učiteľov. Návrh etického kódexu učiteľa. dostupné na internete `http://www.komoraucitelov.org/clanok-39-navrh-etickeho-kodexu-/ucitela.html/`, 2010. pristúpené 19-December-2013.

Digman, J. M. 1990. Personality structure: Emergence of the five-factor model. Annual Review of Psychology, 41, 417-440.

Popkins, N. C. The Five-Factor Model: Emergence of a Taxonomic Model for Personality Psychology. dostupné na internete `http://www.personalityresearch.org/papers/popkins.html`, 1998. pristúpené 23-Január-2014.

Beaumont L. R. Five Factor Constellations and Popular Personality Types. dostupné na internete http://www.emotionalcompetency.com/papers/FiveFactor-ConstellationsandPopularpersonalitytypes.pdf, 2003. pristúpené 23-Január-2014.

Cherry, K. The Big Five Personality Dimensions - Overview of the Big Five Personality Dimensions. dostupné na internete http://psychology.about.com/od/personalitydevelopment/a/bigfive.htm, 2008. pristúpené 26-Január-2014.

Dr. Randall H. Lucius, Ph.D. The ABC's of establishing a good hiring process. dostupné na internete http://news.fitability.com/core/item/page.aspx?s=17622.0.44.24, 2003. pristúpené 26-Január-2014.

Exploring Computer Science. CS Education statistics. dostupné na internete http://www.exploringcs.org/resources/cs-statistics, 2013. pristúpené 27-Január-2014.

Šebejová, D.: Automatická kontrola postupu študenta v prípadových štúdiách pre výučbu objektového programovania. Diplomová práca, 2012, Technická univerzita v Košiciach.

Zoznam príloh

Príloha A Systémová príručka

Príloha B Používateľská príručka

Príloha C Prehľad nástrojov projektovej komunikácie

Príloha D Odborný článok k diplomovej práci

Príloha E CD príloha obsahujúca záverečnú prácu v elektronickej podobe