Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Podpora dištančného vzdelávania v predmete Systémové programovanie a asemblery

Tímový projekt

(Dokumentácia k riadeniu projektu –letný semester)

Členovia tímu:

Bc. Ľudovít Fülöp

Bc. Martin Lacko

Bc. Ivan Malich

Bc. Dalimír Orfánus

Bc. Ivan Straka

Pedagogický vedúci:

Doc. Ing. Pavel Čičák, PhD.

2003/2004 Tím č.10

Obsah

Prílohy:

1 Úvod

Tento dokument sa zaoberá systémom riadenia činnosti tímu č.10 DagwooD na tímovom projekte s názvom Podpora dištančného vzdelávania v predmete Systémové programovanie a asemblery. Je súhrnom informácií a dokumentov vypracovaných v rámci predmetu Riadenie projektov v informatike.

2 Riadenie projektu

2.1 Plán projektu na letný semester

Plán projektu bol navrhnutý na začiatku semestra. Predstavoval určitú zmenu oproti návrhu zo zimy, keďže došlo aj ku zmene špecifikácie.

Plán navrhol: Dalimír Orfánus

Shválil: celý tím.

- pôvodný plán.

- skutočnosť.

	týždeň												
Úlohy	1	2	3	4	5	6	7	8	9	**10**	11	12	**13**
návrh modulov													
implementácia modulov													
testovanie modulov													
integrácia modulov													
testovanie celku													
používanie a údržba													
dokumentácia modulov													
dokumentácia celku													

V 10. a 13. týždni boli kontrolné body.

2.2 Úlohy v letnom semestri

Úlohy sa odvodzujú zjemňovaním plánu projektu na každý týždeň a od stavu plnenia úloh v predchádzajúcom týždni.

Zoznam úloh vypracoval a udržiaval aktuálny: Dalimír Orfánus

Prideľovanie úloh: po diskusii tím.

Stav úlohy:

- = splnená
- = aktívna (pracuje sa na nej)
- = pokračuje (nebola dokončená, prípadne bola rozšírená a pokračuje v ďalšom týždni)

= zrušená

riešiteľ	kód	stav	úloha
			Úlohy na 13. týždeň letného semestra (do 17. 05. 2004)
všetci	21.1		Technická dokumentácia.
Fülöp	21.4		Podpora hromadného pridávania používateľov do Moodle.
Lacko	21.3		Používateľská príručka produktu.
Malich	23.1		Inštalačná príručka produktu.
	25.1		Pripraviť statickú prezentáciu projektu.
Orfánus	17.7		Správa dokumentácie.
	5.4		Správa a aktualizácia web stránky.
Straka	21.4		Podpora hromadného pridávania používateľov do Moodle.
			Úlohy na 12. týždeň letného semestra (do 10. 05. 2004)
všetci	21.1		Technická dokumentácia.
Fülöp	21.4		Podpora hromadného pridávania používateľov do Moodle.
	20.5		Upraviť Moodle na podporu osobných čísel.
Lacko	21.3		Používateľská príručka produktu.
Malich	23.1		Inštalačná príručka produktu.
	22.2		Príprava inštalačného skriptu.
	25.1		Pripraviť statickú prezentáciu projektu.
Orfánus	17.7		Správa dokumentácie.
	5.4		Správa a aktualizácia web stránky.
Straka	21.4		Podpora hromadného pridávania používateľov do Moodle.
	20.5		Upraviť Moodle na podporu osobných čísel.
			Úlohy na 11. týždeň letného semestra (do 03. 05. 2004)
všetci	21.1		Technická dokumentácia.
Fülöp	21.4		Podpora hromadného pridávania používateľov do Moodle.
Lacko	21.3		Používateľská príručka produktu.
Malich	23.1		Inštalačná príručka produktu.

	22.2		Príprava inštalačného skriptu.
Orfánus	19.1		Rozšírenie Moodle (Kalendár, FAQ, Udalosti)
	19.2		Návrh a implementácia perspektív (pohľadov) v Moodle. (Dokončiť manažovanie poriadia uzlov na rovnakej úrovni)
	17.7		Správa dokumentácie.
Straka	5.4		Správa a aktualizácia web stránky.
	21.4		Podpora hromadného pridávania používateľov do Moodle.
	20.5		Upraviť Moodle na podporu osobných čísel.

Úlohy na 10. týždeň letného semestra (do 26. 04. 2004)

všetci	21.1		Technická dokumentácia.
Fülöp	21.3		Používateľská príručka produktu.
	21.4		Podpora hromadného pridávania používateľov do Moodle.
Lacko	21.3		Používateľská príručka produktu.
Malich	23.1		Inštalačná príručka produktu.
	23.2		Pripraviť CD k odovzdaniu.
	22.1		Doladenie démona pre testovanie funkčnosti
	22.2		Príprava inštalačného skriptu.
	23.3		Vytvoriť kurz Používateľská príručka na našom Moodle pre doc. Čičáka.
	23.4		Kontrola a úprava príručiek.
Orfánus	19.1		Rozšírenie Moodle (Kalendár, FAQ, Udalosti)
	19.2		Návrh a implementácia perspektív (pohľadov) v Moodle.
	17.7		Správa dokumentácie.
Straka	5.4		Správa a aktualizácia web stránky.
	21.4		Podpora hromadného pridávania používateľov do Moodle.
	20.5		Upraviť Moodle na podporu osobných čísel.
	22.2		Príprava inštalačného skriptu.

Úlohy na 8. a 9. týždeň letného semestra (do 19. 04. 2004)

všetci	21.1		Technická dokumentácia.
Fülöp	22.1		Doladenie démona pre testovanie funkčnosti
	21.3		Používateľská príručka produktu.
	21.4		Podpora hromadného pridávania používateľov do Moodle.
	20.1		Rozhranie Moodle k obslúžnemu démonu.

Lacko	21.3		Používateľská príručka produktu.
Malich	22.1		Doladenie démona pre testovanie funkčnosti
	22.2		Príprava inštalačného skriptu.
	22.3		Príprava prezentácie o stave riešenia projektu.
Orfánus	19.1		Rozšírenie Moodle (Kalendár, FAQ, Udalosti)
	19.2		Návrh a implementácia perspektív (pohľadov) v Moodle.
	17.7		Správa dokumentácie.
	5.4		Správa a aktualizácia web stránky.
	21.4		Podpora hromadného pridávania používateľov do Moodle.
Straka	20.5		Upraviť Moodle na podporu osobných čísel.
	22.2		Príprava inštalačného skriptu.

Úlohy na 7. týždeň letného semestra (do 05. 04. 2004)

všetci	21.1		Technická dokumentácia.
	21.2		Podkladový materiál k prezentácii stavu riešenia projektu.
	21.3		Používateľská príručka produktu.
Fülöp	21.4		Podpora hromadného pridávania používateľov do Moodle.
	20.1		Rozhranie Moodle k obslúžnemu démonu.
	20.2		Implemnetácia ADT strom a jeho GUI.
Lacko	20.2		Implemnetácia ADT strom a jeho GUI.
	21.5		Úprava bezpečnosti v skripte.
Malich	17.5		Návrh a implementácia démona pre správu a menežment odovzdaných zadaní.
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
	21.6		Pripraviť vzor na písanie podkladov k prezentácii o stavu riešenia projektu.
Orfánus	19.1		Rozšírenie Moodle (Kalendár, FAQ, Udalosti)
	19.2		Návrh a implementácia perspektív (pohľadov) v Moodle.
	17.7		Správa dokumentácie.
	5.4		Správa a aktualizácia web stránky.
Straka	20.4		Zistiť spôsob generovania hesiel na FIIT, analyzovať a navrhnúť spôsob hromadného registrovania a zapisovania používateľov.
	20.5		Upraviť Moodle na podporu osobných čísel.

Úlohy na 6. týždeň letného semestra (do 29. 3. 2004)

Fülöp	18.1		Úpravy v Moodle na podporu viac verzií odovzdaných súborov.

	20.1		Rozhranie Moodle k obslúžnemu démonu.
	20.2		Implemnetácia ADT strom a jeho GUI.
Lacko	16.2		Tvorba rozhrania modulu plagiátorstva (zaobalenie do SOAP protokolu).
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
Malich	17.5		Návrh a implementácia démona pre správu a manažment odovzdaných zadaní.
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
Orfánus	20.3		Model údajov v Moodle.
	19.1		Rozšírenie Moodle (Kalendár, FAQ, Udalosti)
	19.2		Návrh a implementácia perspektív (pohľadov) v Moodle.
	17.7		Správa dokumentácie.
	5.4		Správa a aktualizácia web stránky.
Straka	16.10		Modul kontroly funkčnosti: spojazdniť existujúce riešenia a upraviť pre naše použitie (zaobaliť ich do SOAP protokolu).
	20.4		Zistiť spôsob generovania hesiel na FIIT, analyzovať a navrhnúť spôsob hromadného registrovania a zapisovania používateľov.
	20.5		Upraviť Moodle na podporu osobných čísel.
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.

Úlohy na 5. týždeň letného semestra (do 22. 3. 2004)

Fülöp	18.1		Úpravy v Moodle na podporu viac verzií odovzdaných súborov.
Lacko	16.2		Tvorba rozhrania modulu plagiátorstva (zaobalenie do SOAP protokolu).
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
Malich	17.5		Návrh a implementácia démona pre správu a menežment odovzdaných zadaní.
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
Orfánus	19.1		Rozšírenie Moodle (Kalendár, FAQ, Udalosti)
	19.2		Návrh a implementácia perspektív (pohľadov) v Moodle.
	17.7		Správa dokumentácie.
	5.4		Správa a aktualizácia web stránky.
Straka	16.10		Modul kontroly funkčnosti: spojazdniť existujúce riešenia a upraviť pre naše použitie (zaobaliť ich do SOAP protokolu).
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.

Úlohy na 4. týždeň letného semestra (do 15. 3. 2004)

Fülöp	17.2		Návrh novej XML-based databázy.
	18.1		Úpravy v Moodle na podporu viac verzií odovzdaných súborov.
Lacko	16.2		Tvorba rozhrania modulu plagiátorstva (zaobalenie do SOAP protokolu).

	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
Malich	17.5		Návrh a implementácia démona pre správu a menežment odovzdaných zadaní.
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
	18.2		Nájsť, resp. navrhnúť XML editor a jeho integráciu do Moodle.
Orfánus	17.7		Správa dokumentácie.
	5.4		Správa a aktualizácia web stránky.
Straka	16.10		Modul kontroly funkčnosti: spojazdniť existujúce riešenia a upraviť pre naše použitie (zaobaliť ich do SOAP protokolu).
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.

Úlohy na 3. týždeň letného semestra (do 8. 3. 2004)

	17.1		Analýza realizácie bázy znalostí v Moodle.
Fülöp	17.2		Návrh novej XML-based databázy.
	17.3		Naštudovať transformáciu XML->HTML (XSLT).
Lacko	16.2		Tvorba rozhrania modulu plagiátorstva (zaobalenie do SOAP protokolu).
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
	16.4		Spustiť vybraný systém pre správu verzií nad našou testovacou a vývojou verziou projektu Moodle.
Malich	17.5		Návrh a implementácia démona pre správu a menežment odovzdaných zadaní.
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.
	17.6		Nasadenie Liberty Alliance.
Orfánus	17.7		Správa dokumentácie.
	5.4		Správa a aktualizácia web stránky.
Straka	16.10		Modul kontroly funkčnosti: spojazdniť existujúce riešenia a upraviť pre naše použitie (zaobaliť ich do SOAP protokolu).
	17.4		Návrhu protokolu pre prenos našich údajov nad SOAP.

Úlohy na 2. týždeň letného semestra (do 1. 3. 2004)

Fülöp	16.1		Analyzovať odovzdávanie súborov v systéme Moodle a možnosti prispôsobenia pre naše ciele („spool daemon“).
Lacko	16.2		Tvorba rozhrania modulu plagiátorstva (zaobalenie do SOAP protokolu).
	16.3		Naštudovať a spojazdniť projekt SourceID (JAVA implementácia projektu Liberty Alliance).
Malich	16.4		Spustiť vybraný systém pre správu verzií nad našou testovacou a vývojou verziou projektu Moodle.
	16.5		Stretnutie s konkurenčným tímom a oficiálne ich oboznámiť s našou ponukou na spoluprácu.
Orfánus	16.5		Stretnutie s konkurenčným tímom a oficiálne ich oboznámiť s našou ponukou na spoluprácu.
	16.6		Úprava a vizualizácia plánu na letný semester.
	16.7		Štábna kultúra na písanie dokumentácie riešenia projektu.

	16.8		Štábna kultúra na písanie záznamov do „changelog“ súborov (pre moduly a súbory so zdrojovým kódom).
	16.9		Vytvoriť hlavičky do zdrojových súborov,
	5.4		Správa a aktualizácia web stránky.
Straka	16.10		Modul kontroly funkčnosti: spojazdniť existujúce riešenia a upraviť pre naše použitie (zaobaliť ich do SOAP protokolu).

2.3 Zodpovednosť za jednotlivé časti

Pre každú časť dokumentácie je uvedený jej realizátor, v prípade, že na úlohe spolupracovalo viacero členov tímu, je na prvom mieste uvedená zodpovedná osoba.V nasledujúcej tabuľke je uvedený realizátor tej ktorej časti odovzdanej dokumentácie tretej etapy.

Časť dokumentácie 3. etapy	**Realizátor**
7. Riešenie	Dalimír Orfánus
7.1 Ohraničenia, zmeny špecifikácie, priority riešenia	Dalimír Orfánus
7.1.1 Zmena z pohľadu systému	Dalimír Orfánus
7.1.2 Báza konceptov	Dalimír Orfánus
7.1.3 Správa používateľov	Dalimír Orfánus + Ivan Straka
7.1.4 Odovzdávanie zadaní	Ľudovít Fülöp
7.1.5 Testovanie vedomostí	Ľudovít Fülöp
7.1.6 Démon	Ivan Malich
7.1.7 Testovanie funkčnosti a plagiátorstva	Dalimír Orfánus
7.2 Rozdelenie projektu	Dalimír Orfánus
7.3 Báza koncpetov	Dalimír Orfánus
7.4 Zisťovanie plagiátorstva	Martin Lacko
7.5 Testovanie funkčnosti	Ivan Straka
7.6 Správa používateľov	Ivan Straka
7.7 Odovzdávanie zadaní	Ľudovít Fülöp
7.8 Démon	Ivan Malich
8 Testovanie systému	Dalimír Orfánus
8.1 Testovanie modulu správy zadaní	Ľudovít Fülöp
9 Čo sme sa naučili a čo nestihli	Dalimír Orfánus
10 Používateľská príručka	Martin Lacko + Dalimír Orfánus
11 Inštalačná príručka	Ivan Malich
12 Zhodnotenie	Dalimír Orfánus
Príloha D	Dalimír Orfánus + Ľudovít Fülöp

3 Posudky a reakcie na posudky

Kapitola zhromažduje dokumenty, ktoré vznikli ako posudky prípadne reakcie na projekt. Je združná z posudku na prototyp od tímu č.11, ktorý je konkurenčným tímom pri riešení zadanej témy. Okrem posudku je tu aj reakcia na neho.

Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Posudok k prototypu tímu Seagles od tímu Dagwood

Tímový projekt

2003/2004

Dagwood (tím č. 10):

Bc. Ľudovít Fülöp

Bc. Martin Lacko

Bc. Ivan Malich

Bc. Dalimír Orfánus

Bc. Peter Procházka

Bc. Ivan Straka

Seagles (tím č. 11):

Bc. Juraj Varíny

Bc. Juraj Prokša

Bc. Andrej Slamka

Bc. Peter Grodovský

Bc. Andrej Kováčik

Bc. Attila Štrba

Posudzovali sme ukončený vývoj prototypu systému na podporu dištančného vzdelávania v predmete Systémové programovanie a asemblery, ktorý je vytváraný v rámci predmetu Tímový projekt na FIIT STU tímom číslo 11, v akademickom roku 2003/2004.

Prezentácia bola zverejnená na adrese: http://labss2.dcs.elf.stuba.sk/TeamProject/2003/team11/prototyp/, posudzovaná ku dňu 19. 12. 2003.

Funkčná stránka prototypu

Grafické spracovanie prototypu i spôsob ovládania je podobný ako u iných (i komerčných) systémoch s podobnou funkciou. Možnosti práce editora spĺňajú požiadavky, ktoré si tím stanovil implementovať v prototype. Súčasťou editora učebných materiálov je i editor pre HTML kód "ekit.jar" (www.hexidec.com), ktorý však nikde v dokumentácii nie je spomenutý. Aj keď bol kód editora prevzatý z iného projektu, považujeme jeho použitie za veľmi pozitívne.

Za miernu nevýhodu prototypu považujeme pomalosť systému. Keďže sa však jedná iba o prototyp, predpokladáme, že vo finálnej verzii bude tento problém odstránený. Ako však uvádzajú samotní autori, rozhodli sa rýchlosť celého systému vymeniť za platformovú nezávislosť.

Vzhľad prototypu

Ako sme už uviedli, vytvorený prototyp využíva zaužívané používateľské rozhranie. Výber tohto prístupu umožnil jednoduchšiu a intuitívnejšiu prácu so systémom, keďže používateľ si nemusí zvykať na nové ovládacie prostredie. Jednoduchosť používania vyplýva z implementovania štandardných ovládacích prvkov využívaných pri bežnej práci s PC.

Páčilo sa nám celkové farebné zladenie prostredia. Celkový vzhľad bol podmienený aj použitím technológie Java. Tento postup mal na výzor prototypu výrazný účinok, preto bol náš dojem zo vzhľadu prototypu pozitívny.

Dokumentácia k prototypu

Samotná používateľská príručka k prototypu je miestami až príliš stručná, autori zjavne predpokladajú skúsenosti administrátora systému s inštaláciou a prevádzkou podobných systémov. V dokumentácii k finálnej verzii by ale nezaškodilo podrobnejšie opísať požiadavky implementovaného systému na operačný systém a na jeho nainštalované súčasti (nainštalovanie JDCB ovládača pre MySQL nie je spomenuté ani v požiadavkách (kapitola 4.4.1)). V kapitole 4.7, v časti "Úprava údajov v predmete" je napísané: "Spustite editor. Vyberte predmet. ...". Bolo by vhodné uviesť aspoň názov spúšťacieho súboru.

I keď sme na používanie slov "užívateľ" a "používateľ" upozorňovali už v minulom posudku, sú v dokumentácii k prototypu použité obe tieto slová, napr. v obr. 10 (str. 36) je slovo užívateľ, na str. 40 je používatel. V texte dokumentácie sa nachádza aj niekoľko pravopisných chýb ale aj formálnych, na str. 37 (kap. 4.2) chyba "skryptovací"; str. 42 v texte je odkaz na obr. 1 (zjavne má byť 14) a chyba v rozpise názvu tímu (Sutions...) obr. 13 - generátor stránok.

CD priložené k dokumentácii

Príjemným prekvapením bolo CD, ktoré bolo priložené k dokumentácii k prototypu. CD okrem samotného prototypu obsahovalo i súbory internetovej stránky tímu a taktiež ovládače JDCB potrebné na komunikáciu s MySQL. Škoda, že CD neobsahovalo i inštalačný manuál k tomuto ovládaču, ktorý sa s ním ale bežne dodáva. Toto CD, žiaľ, nie je v dokumentácii spomenuté, chýba najmä popis obsahu CD. Prototyp, ktorý je uloženený na CD, však nie je možné bez úprav v zdrojových súboroch spustiť na počítači bez pripojenia na server labss2. Súbory generátora stránok na CD sú pravdepodobne nekompletné, pretože sa v zdrojových PHP súboroch vyskytujú odkazy na súbory, ktoré sa majú nachádzať v adresári /images (ikony vo formáte gif).

Celkové zhodnotenie

Náš celkový dojem z prototypu tímu Seagles je veľmi dobrý. Prototyp podľa nášho názoru spĺňa ciele, ktoré si členovia tímu stanovili – overiť správnosť vybraných častí hrubého návrhu a možnosti jeho implementácie. Ak bude aj ďalšia práca tímu pokračovať v nastolenom trende, veríme, že svoj projekt úspešne ukončia a výsledkom bude plne funkčný produkt.

Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Stanovisko k posudku prototypu od tímu Seagles

Tímový projekt

2003/2004

Autori stanoviska:

Dagwood (tím č. 10):

Bc. Ľudovít Fülöp

Bc. Martin Lacko

Bc. Ivan Malich

Bc. Dalimír Orfánus

Bc. Peter Procházka

Bc. Ivan Straka

Autori posudku:

Seagles (tím č. 11):

Bc. Juraj Varíny

Bc. Juraj Prokša

Bc. Andrej Slamka

Bc. Peter Grodovský

Bc. Andrej Kováčik

Bc. Attila Štrba

Toto je vyjadrenie stanoviska k dokumentu *Posudok prototypu konkurenčného tímu*, ktorí vytvoril tím č.11 – Seagles. Chceme sa poďakovať konkurencii za jej pozorné preštudovanie dokumentácie a otestovanie nášho prototypu. K tvrdeniam, s ktorými sa nestotožňujeme, sa vyjadríme ďalej v tomto dokumente.

Posudok celkovej dokumentácie

Vnímame veľmi pozitívne úsilie tímu Seagles, že si opätovne preštudovali časti dokumentácie, ku ktorým sa už vyjadrili a boli už „uzavreté“. Rozhodli sme sa prepracovať dokumentáciu 1. etapy, pretože sme po jej odovzdaní boli presvedčení, že je ju nutné vylepšiť i za tú cenu, že to už nijako neovplyvní jej hodnotenie. Presvedčenie pramenilo z toho, že chceme našim budúcim nasledovníkom zanechať dobrý a použiteľný zdroj informácií v podobných projektoch.

Posudok dokumentácie k prototypu

„Chýba opis prototypovanej oblasti, opis implementácie prototypu, ako aj podrobnejší popis samotného prototypu.“

Nesúhlasime úplne s týmto tvrdením odvolávajúc sa na kapitolu *6.1 Cieľ prototypovania*, kde sú uvedené primárne ciele prototypovania vrátane oblasti prototypovania. Cieľom prototypovania nebolo otestovať konkrétnu knižnicu (napr. pre SOAP), ale myšlienku použitia týchto technológií. Uznávame však, že sme mohli podrobnejšie popísať samotnú implementáciu prototypu. Z časového hľadiska sa nám to však nepodarilo, keďže išlo o prvé praktické oboznamovanie sa s použitými technológiami.

Posudok prototypu

„Prototyp pravdepodobne nebol testovaný pod prehliadačom pod OS Windows, nakoľko sa vyskytli malé problémy s kódovaním znakov.“

S týmto tvrdením nemôžeme súhlasiť. Ako prehliadač sme používali platformovo nezávislý programový balík Mozilla vo verziách pre rôzne operačné systémy, vrátane OS Windows. Príčinou uvedeného problému bol zle nakonfigurovaný HTTP server, ktorý nerešpektoval meta informáciu o skutočnej znakovej sade HTML dokumentov (v našom prípade išlo o UTF-8), pretože do HTTP hlavičky vkladal údaj o kódovaní v znakovej sade Latin-1. Následné nesprávne zobrazenie diakritiky na www stránkach potom vyplýva z faktu, že prehliadače v takýchto prípadoch akceptujú informáciu z HTTP hlavičky a teda stránky sa zobrazia v inej znakovej sade i napriek meta informácii v HTML dokumente. Navyše na počítačoch, na ktorých sme prezentovali prototyp, nebola znaková sada UTF-8 nainštalovaná. Ak sme teda aj v prehliadači nastavili správnu znakovú sadu, stránky boli stále „zlé“. Tomuto sa však nedalo zabrániť, kedže prototyp bol už odovzdaný pričom na počítačoch, na ktorých sme testovali prototyp všetko fungovalo.

4 Štandardy tímovej práce

Štandardy tímovej práce a dokumnety štábnej kultúry sú doplnkovými dokumentami pri riadení tímového projektu. Jedná sa o doporučenia, ktoré sme dorživali počas práce, aby sme zvýšili efektivitu tímu ako celku.

Podkapitolu 4.1 vypracovali spoločne Ivan Malich a Dalimír Orfánus. Ostatné podkapitoly vypracoval Dalimír Orfánus.

4.1 Komunikácia v tíme

4.1.1 Interná komunikácia

4.1.1.1 Komunikačné kanály a ich priorita

Komunikácia medzi členmi tímu prebieha zväčša neformálnym spôsobom. Dôvodom je to, že tím je založený na priateľských vzťahoch medzi členmi tímu, nie na direktívnych základoch.

Najefektívnejším spôsobom komunikácie je priamy rozhovor členov tímu. Najlepšie to je vidno na pravidelných tímových stretnutiach, kedy si za pomerne krátky čas vyjasníme otázky okolo dokončených a plánovaných prác, ako aj celkový pohľad na ďalšie napredovanie projektu. Zaujímavosťou je v tomto kontexte fakt, že jeden člen nášho tímu s nami spolupracuje zo zahraničia. Priamy rozhovor s ním je možný prostredníctvom videokonferencie.

Ak keď je priamy rozhovor najefektívnejším spôsobom komunikácie, najčastejšie využívame v našom tíme písomnú komunikáciu. Ak sa potrebujeme rýchlo dohodnúť na nejakých veciach, používame na tento účel ICQ. ICQ umožňuje viesť niečo na spôsob rozhovoru v písomnej forme.

Pomerne veľké množstvo informácií nám umožňuje prenášať e-mail. Aby sme zjednodušili komunikáciu prostredníctvom e-mailu, drvivá väčšina e-mailov je posielaná prostredníctvom mailing-listu vytvoreného práve na tento účel. Tým je zabezpečené, že nové informácie s prípadnými prílohami sa dostanú ku všetkým členom tímu, ktorí majú možnosť hneď reagovať. Odpovede potom opäť smerujú všetkým členom tímu.

V prípadoch, ktoré si vyžadujú rýchle odovzdanie informácie, využívame mobilné telefóny. Typicky ide o oznámenie drobného meškania kvôli dopravnej situácii a podobne.

Pri rozdelení komunikácie podľa frekvencie pripadá najväčšia časť komunikácie na elektronickú poštu a ICQ. Týmto spôsobom si vymieňame veľkú väčšinu informáácií a

dokumentácie. Ďalej nasleduje priamy rozhovor na tímových stretnutiach raz za týždeň, kedy prejednávame hlavné rysy ďalšieho napredovania projektu. Komunikáciu pomocou mobilných telefónov využívame pomerne zriedkavo.

Komunikácia pri regulárnom strtnutí má najmä kontrolnú a motivačnú funkciu. Informačná funkcia prevažuje zase pri komunikácii prostrednícvtom elektronickej pošty, ICQ alebo cez telefón.

4.1.1.2 Vnutorná organizácia

Vnútorná organizácia tímu je viditeľná hlavne pri pravidelných tímových stretnutiach a komunikácii tímu s okolím. Ide hlavne o plánovanie projektu a v tejto fáze projektu o návrh produktu. Všetky rozhodnutia však vychádzajú zo spoločnej dohody medzi všetkými členmi tímu, aby bolo možné zohľadniť všetky pohľady členov tímu na produkt a ich názory. Tu sa na základe pridelených úloch tím rozdelil na niekoľko menších nedisjunktných skupiniek. Skupinky sa formujú aj podľa problému, ktorý sa rozoberá, keď sa napr. niektorí pričlenia k názoru jedného a zvyšok k druhému diskutujúcemu členovy. Príklad jednej skupinky: Pri riešení problému vnútorného rozhrania systému je Ivan Malich v skupinke spolu s Martinom Lackom a Ivanom Strakom a súčasne je aj v skupinke Dala Orfánusa a Ľudovíta Fülöpa.

Pri bežnej komunikácii o spolupráci medzi jednotlivými členmi tímu však väčšinou žiadna organizácie nie je viditeľná, riadia si ju vždy tí členovia tímu, ktorí spolu vyvíjajú nejakú časť systému.

4.1.2 Komunikácia s okolím

Komunikáciu s okolím zabezpečujú všetci členovia tímu, keď neformálne komunikujeme napr. s členmi iných tímov. Oficiálnu komunikáciu (stanoviská, referáty a podobne) má na starosti predovšetkým vedúci tímu. Ten prezentuje stanoviská celého tímu (po pripomienkach a dohode tímu) voči ostatému svetu.

Špeciálny úlohu má pedagogický vedúci - doc. Čičák. Priamo sa zúčastňuje na našich regulérnych stretnutiach, takže je súčasťou procesu vývoja projektu. Okrem sledovania prác na projektu hodnotí tím z rôznych aspektov ako napr. súdržnosť a pod.

4.1.3 Zdieľanie informácií a zdrojových kódov

Táto časť sa čiastočne svojim obsahom prekrýva s 1. časťou tohto dokumentu, pretože sem okrem zdieľania zdrojových kódov spadá aj vytváranie dokumentácie a čiastočne aj iná komunikácia. Keďže komunikácia už bola všeobecne popísaná, ťažisko tejto časti bude na vytváraní dokumentácie a samotného produktu.

4.1.3.1 Vytváranie dokumentácie

Za vytváranie dokumentácie nesie yodpovednosť každý člen. Každý má na starosti implementáciu inej súčasti systému, ku ktorej vytvára dokumentáciu návrhu, programátorskú príručku a systémovú príručku. Na dokumentáciu návrhu používame nami vytvorený vzor, aby celková dokumentácia návrhu prodoktu pôsobila jednotne a bola písaná jednotným štýlom. Programátorská príručka obsahuje popis všetkých implementovaných funkcií systému a rozhraní smerom k iným častiam produktu. Táto dokumentácia slúži hlavne na bezproblémové spojenie všetkých implementovaných súčastí produktu do jedného funkčného celku a je vytváraná priebežne. Iné časti dokumentácie (používateľská príručka, systémová príručka) nie sú vytvárané priebežne, pretože jej obsah úzko súvisí s hotovým produktom a preto sa začne vytvárať až pri finalizácii celého systému.

4.1.3.2 Zdieľanie zdrojových kódov

Typickým problémom pri tímovej práci je zdieľanie vytvorených súčastí jednotlivými členmi tímu. Pri programátorských prácach je to najčastejšie riešené pomocou nástroja CVS, ktorý umožňuje centrálne uloženie zmien od jednotlivých autorov.

My sme sa rozhodli namiesto CVS použiť nástroj GNU arch. Jeho použitie je v princípe rovnaké ako pri CVS ale má aj takú funkcionalitu, ktorú pomocou CVS nie je možné dosiahnuť. Medzi hlavné výhody patrí to, že nie je potrebné udržiavať žiadny centrálny server. Ako úložisko môže slúžiť obyčaný adresár s príslušnou štruktúrou podadresárov a súborov. Prístup do tohto úložiska je možný lokálne (vyššiea rýchlosť oproti CVS) alebo vzdilene (cez HTTP, FTP, SFTP).

Popri jednému centrálnemu úložisku má každý člen tímu aj svoje vlastné, kde vyvíja svoju časť systému. Keď je spokojný s vlastnosťami, sprístupní svoju prácu ostatným členom tímu prostredníctvom centrálneho úložiska alebo cez e-mail (hlavne ak ide o dokumentáciu).

4.1.3.3 Sledovanie vývoja projektu

Aby bolo možné sledovať vývoj celého projektu, každý člen tímu si zapisuje všetky úkony, ktoré pri vývoji vykonal. Na tento účel sme zaviedli changelogy. Všetky changelogy sú formátované podľa jednotného štýlu (formát changelogu je XML, definície sú v DTD, Príloha A), takže je ich možné v prípade nutnosti konvertovať do ľubovoľného tvaru. Máme definované dva druhy changelogov. Jeden je pre sledovanie zmien v rámci modulu. Zaznamenávame do neho všetky zmeny v zdrojovych súboroch, ktoré sa týkaju modulu. Druhý changelog slúži pri sledovanie

vývoju celkového produktu. Každý modul má vlastný strom, do ktorého sa robia zázznamy v prípade zmeny funkcionality navonok. Teda v prípade vnútorných zmien, ktoré nezmenia funkcionalitu modulu, tieto zmeny sa nezaznamenávajú.

4.2 Stav projektu (šablóna)

Tento vzor vám má poslúžiť na rýchlejšie vypracovanie podkladového materiálu k prezentácii stavu projektu, ale aj pre vnútornu kontrolu tímu. Dokument povinne vypracuje každý člen tímu. V prípade, že bolo členovi pridelených viac ako jedna dlhodobá úloha, vypracuje v príslušnom rozsahu dokument ku každej z nich. Tento dokument bude súčasťou dokumentácie riadeniu projektu. Účelom dokumentu o stave riešenia projektu nie je podať podrobnú a výčerpávajúcu dokumentáciu návrhu.

4.2.1 Názov modulu (resp. dlhodobej úlohy)

Opíše sa tu účel modulu, akú má mať funkcionalitu. Netreba ísť do detailov, ale treba nejako uviesť do problému. Vyjadrite sa, aký model životného cyklu ste pri práci na module použili (predpokladám, že všetci budete písať evolučný ☺).

4.2.2 Zmena špecifikácie oproti návrhu

V prípade ak došlo k zmene špecifikácie, ktorá bola uvedená v technickej dokumentácii za zimný semester, uvedie sa tu.

4.2.3 Opis návrhu modulu

Opíšete návrh modulu ale nie v detailoch. Detaily sa môžu uviesť v prípade, že riešenie je niečim zvláštne a výnimočné príp. stojí za zmienku. Uvedené tu budú iba základné a stručné fakty. Akú stratégiu návrhu ste použili a prečo (zhora nadol, zdola nahor).

Vyjadrite sa aj k dôležitým návrhovým rozhodnutiam.

4.2.4 Opis postupu implementácie

Ako sa postupovalo(-je) pri implementácii, t.j. zhora-nadol alebo zdola nahor a prečo. Aké technické problémy sa museli vyriešiť a čo sa pri nej použilo (napr. nainštalovať špeciálnu knižnicu a pod.). Dôležité implementačné rozhodnutia, použitý prog. jazyk. Čo všetko sa už implementovalo a aj koľko ešte treba doimplementovať vrátane vášho odhadu. Bude veľmi dobré, ak uvediete, koľko času ste venovali implementácii tohto modulu.

4.2.5 Testovanie

Ako sa vykonávalo testovanie. Či sa testoval ako biela alebo čierna skrinka.

4.2.6 Záver

Tu sa vyjadrite váš vlastný názor k stavu modulu a aj to, čo isto chcete, aby sa na prezentácii spomenulo.

Vypracoval: Dalimír Orfánus

Schválil celý tím.

4.3 Technická dokumentácia (šablóna)

Najprv dajte nejaké všeobecné veci a potom popis funkcie modulu v stručnosti (jeden odstavec). Ak sa zmenila mierne špecifikácia modulu oproti tomu, čo máme v dokumentácii, tak to dajte ako podkapitolu napr. Zapracovanie nedostatkov špecifikácie, ale to hadam nebude treba. Toto sa potom bude musieť napísať v dokumentácii celku.

Nezabudnite to tak navrhnúť, aby bolo možné modul nejakým spôsobom testovať a debugovať. Teda ak niekto sa k tomu časom dostane, aby si po prečítaní tohto dokumentu vedel, ako si môže odskúšať modul a ako overiť, či funguje , alebo nie.

Tak isto nezabudnite na to, že k modulu budú pristupovať rôzne kategórie používateľov. Pre nich treba potom zvlášť mať príslušné politiky. Nezabudnite ani na to, že v návrhu máme zachytenú možnosť sa pripojiť k modulu zvlášť ako administrátor.

Veľmi dôležitá je aj rozširovateľnosť modulu, teda musite to navrhnúť tak, aby sa dala funkcionalita pomerne jednoducho rozširovať definovaným spôsobom.

Ak to bude stat za zmienku, napíšte malú analýzu použitej knižnice. V prípade, že vytvoríte vlastnú, nezabudnite ju dokumentovať, aby sa to dalo do príručky používania (systémovej).

Ja sa pokusím ešte vygenerovať štábnu kultúru na písanie týchto príručiek, ale to sa budem musieť najprv poradiť s vami a bude to zaujímavé v cca 7. týždni, keď začne testovanie.

4.3.1 Architektúra modulu

Ak modul tvoria nejaké logické celky, tak ako sú prepojené, ako prebieha komunikácia a samozrejme popísať. Treba si uvedomiť, že toto je podroný návrh, teda ak si to niekto zobere do ruky, tak mal hneď vedieť ako implementovať.

4.3.2 Fyzický model údajov

Tu hadam netreba nič vysvetlovať. Nezabudnite, že diagram bez popisu nie je žiadny diagram... Prípadne ako to robíte objektovo, bolo by vhodné, ak by tu bol aj digram tried (v samostatnej kapitole).

4.3.3 Diagram tokov údajov

To isté ako Fyzický model údajov ☺. Nie je to však nevyhnutné. Keď to bude, bude to super.

4.3.4 Algoritmy spracovania

Keďže funkcionalita modulu je známa zo špecifikácie, tak sa ukážu, aké algoritmy sa použili a popíšu sa!!! V prípade potreby môžem niečo prekresliť do statechartov, ak budem stíhať. Teda pre každý algoritmus aspoň jeden odstvec ak nie rovno podkapitolu ☺.

4.3.5 Opis realizácie

Tu opíšete, ako ste to kódovali, čo sa muselo zmeniť, nastaviť, ako ste postupovali. Čo ste implementovali najskôr a prečo a pod. Však je to z názvu jasné ☺.

4.3.6 Overenie výsledku

Myslí sa testovanie. Návrh, ako testovať modul, aké testovacie vzorky použiť, ako overiť, že výsledok testovania je OK.

Vypracoval: Dalimír Orfánus

Schválil celý tím.

4.4 Štandardy kódovania

Účelom tohto dokumentu je definovať jednotnú štruktúru a vzhľad zdrojových súborov (štábnu kultúru), ktoré budú vytvorené počas implementácie tímom DagwooD v predmete Tímový projekt. Uvedené zásady je nevyhnutné dodržiavať aby sa zachovala čitateľnosť programov.

4.4.1 Hlavička súborov

Každý vytvorený súbor bude mať povinne hneď na začiatku pred kódom túto hlavičku:

```
/**
 * @FILE       : nazov_suboru.php
 * @MODULE     : nazov modulu, ktory ho pouziva
```

```
 * @MODULE     : muoze byt aj viac krat
 *
 * @CURRENT_VER  : sucasna verzia
 * @PREVIOUS_VER : predchadzajuca verzia
 *
 * @AUTHOR: meno <meno@...>
 * @AUTHOR: meno <meno@...>
 *
 * @REMARKS: Poznamky k suboru a implemntacii.
 *
*/
```

4.4.2 Štábna kultúra – PHP

4.4.2.1 Odsadzovanie textu

Text sa bude odsadzovať použitím tabelatóra s dĺžkou 4 znaky.

4.4.2.2 Zložené príkazy

Zložené príkazy sa budú zapisovať tak, že otvárací symbol '{' sa bude nachádzať v tom istom riadku za príkazom, ku ktorému patrí. Uzatvárcí symbol '}' bude začínať na novom riadku a bude tak odsadený, aby bol zarovno s príkazom, ku ktorému patrí. Príklad:

```
for($i = 0; $i < 10; $++) {
      prikaz1;
      prikaz2;
      ...
}
```

4.4.2.3 Názvy premenných

Názvy premenných budú anglické a vždy začínať malým začiatočným písmenom. Ak bude názov premennej tvoriť viac slov, spoja sa podčiarkovníkom. Napr.: `$count_of_visitors`. Názvy musia byť zvolené tak, aby nahovárali aký účel plní premenná. Toto sa netýka premenných, ktoré sú všeobecne známe a používanie na indexovanie (i, j, k a pod.).

4.4.2.4 Deklarácia triedy

Každá trieda bude deklarovaná podľa nasledujúceho vzroru:

```
class soap_server extends soap {
    /**
     * Strucny popis clenskej premennej
     *
     */
     var $member1;

    /**
```

```
     * Strucny popis funkcie
     *
     * @param string    $arg1
     * @param int       $arg2
     * @return          float
     */
    function foo($arg1, $arg2) {
        global $global_var_1;
        global $global_var_2;
        $var1;
        $var2;

        for($i = 0; $i < $global_var_1; $i++) {
            prikaz1;
            prikaz2;
            ...
            prikazN;
        }

        return $val;
    }

}
```

Pri metódach sa najprv uvedie zoznam premenných, ktoré sa preberajú z globálneho menného priestoru. Potom nasleduje deklarácia lokálnych premenných. Medzi deklaráciou premenných a samotným exekučným telom funkcie sa vynechá aspoň jeden riadok. Premenné nikdy nedeklarujeme v exekučnom tele funkcie. Všetky používané premenné musia byť uvedené hneď na začiatku funkcie. Výnimku môžu tvoriť premenné používané na indexovanie (`$i`, `$j`, `$k` a pod.) za podmienky, že sa nezníži čitateľnosť kódu.

4.4.2.5 Volanie funkcií

Funkcie sa nesmú volať s medzerami medzi menom funkcie a otváracim symbolom pre argumenty '('. Medzery sa naopak musia vložiť pred a za priraďovací symbol '='. Odporúča sa použiť jednu medzeru. V prípade, že to zvýši čitateľnosť, môže sa vložiť viac medzier. Príklad volania:

```
$foo = bar($arg1, $arg2);
```

4.4.2.6 Názvy konštánt

Názvy konštánt budu písané veľkými písmenami. V prípade, že je nejaká konštanta deklarovaná v nejakej triede, uvedie sa najpr meno triedy. Názvy budú vždy v anglickom jazyku. Ak názov tvorí viac slov, použije sa podčiarkovník. Ukážka: `SOAP_SERVER_LISTEN_PORT`.

4.4.2.7 Komentáre

Budeme rozlišovať dva druhy komentárov:

- dokumentačné a
- v riadku (inline).

Dokumentačné komentáre budú uvádzané takouto postupnosťou symbolov: „`/**`“ a ukončené budú „`*/`“. Príklad:

```
/**
 * Toto je priklad dokumentacneho komentara.
 */
```

Tieto komentáre je možné za použitia vhodného externého programu použiť aj na vygenerovanie časti dokumentácie. Príkladom môže byť dokumentačný program JavaDoc alebo PHPDoc.

Komentáre v riadku (inline) budú slúžiť na vysvetlenie algorytmov. Ukážka na blokový „inline“ komentár:

```
/*
 * Priklad na inline komentar.
 */
```

Riadkový „inline“ komentár sa bude zapisovať takto:

```
// Toto je priklad na riadkovy inline komentar
```

4.4.2.8 Vkladanie kódu

Kód sa bude vkladať výlučne použitím funkcie `include_once()` alebo `require_once()`. Výber funkcie záleží od účelu.

5 Role členov tímu

Oproti zimnému semestru odišiel z tímu Bc. Peter Procházka (vedúci tímu, iniciátor, vyšetrovateľ, ťahúň, architekt návrhu, správca dokumentácie I. etapy, správca dokumentácie riadenia, grafika, analytik). Úlohu vedúceho prebral po ňom Bc. Ivan Malich. Správcom dokumentácie III. etapy a riadenia sa stal Bc. Dalimír Orfánus. Úlohy vrtáka sa v letnom semestri zhostil Bc. Ľudovít Fülöp.

Príloha A - DTD na zaznamenávanie zmien

Aby sme mohli jednoduchšie a čo najrýchlejšie zdokumentovať životný cyklus podsystému a celého systému sme vytvorili vlastné DTD podľa ktorého sme robili záznamy do changelog súborov. Na základe týchto súborov vieme nájsť rýclejšie prípadnú príčinu chyby.

Autor prvej verzie: Bc. Dalimír Orfánus.

Modifikácie a návrh zmien (verzia 2): Bc. Ivan Malich.

Finálna podoba a úprava (verzia 3): Bc. Dalimír Orfánus.

A.1 Posdydtém (modul)

```
<!--

  version 3

-->
  <!ELEMENT file_changelog (file*)>
  <!ELEMENT file (description, record*)>
  <!ELEMENT description (#PCDATA)>
  <!ELEMENT record (change*)>
  <!ELEMENT change (#PCDATA)>
  <!--
       filename     : nazov suboru, ktoreho sa tyka tento
changelog
       module       : ktoreho, prip. ktorych modulov sa to tyka
       language     : programovaci jazyk
       since        : kedy sme ho vytvorili, teda zacali prace na
implementacii
       finished     : kedy sa definitivne ukoncili prace na nom -
t.j. zme to zastavili, pripadne sme ho zrusili
  -->
  <!ATTLIST file
            filename      CDATA   #REQUIRED
            module        CDATA   #REQUIRED
            language      CDATA   #REQUIRED
            since         CDATA   #REQUIRED
            finished      CDATA   #IMPLIED
  >
  <!--
       when     : kedy sa vytvoril tento zaznam.
       curr_ver : nova (sucasna erzia)
       prev_ver : z ktorej ver. vznikla sucasna verzia
       who      : kto tento zaznam vytvoril(iba 1 z 5 moznosti ,
t.j. je zodpovedny sa spravne udaje v nom
  -->
  <!ATTLIST record
            when      CDATA   #REQUIRED
```

```
            curr_ver  CDATA   #REQUIRED
            prev_ver  CDATA   #REQUIRED
            who (dalxo | ico | chrco | hollub | kexo) #REQUIRED
  >
  <!--
       desc: kratky popis zmeny. Ak nestaci, muoze sa viac popisat
v tele elementu (vid priklad).
          Kratky popis je naschval povinny, pretoze musi byt jasne
a strucne povedane, co sa zmenilo (pri rychlom vyhladavani).
          Ostatne ako napr. implementacne deatily a vykecavacky
puojdu do toho tela elementu feature (ak treba).
  -->
  <!ATTLIST change
            desc  CDATA   #REQUIRED
  >
```

A.2 Posdydtém (modul)

```
<!--

   version 3

-->
  <!ELEMENT module_changelog (module*)>
  <!ELEMENT module (description, record*)>
  <!ELEMENT description (#PCDATA | file)*>
  <!ELEMENT record (feature*, not_affected_file*)>
  <!ELEMENT feature (#PCDATA | affected_file)*>
  <!ELEMENT affected_file (#PCDATA)*>
  <!ELEMENT not_affected_file (#PCDATA)*>
  <!ELEMENT file (#PCDATA)*>


  <!--
       module_name  : nazov modulu
       since        : kedy sme ho vytvorili, teda zacali prace na
implementacii
       finished     : kedy sa definitivne ukoncili prace na nom -
t.j. zme to zastavili, pripadne zrusili
  -->
  <!ATTLIST module
            name          CDATA   #REQUIRED
            since         CDATA   #REQUIRED
            finished      CDATA   #IMPLIED
  >

  <!--
       when       : kedy pribudol tento zaznam
       curr_ver   : nova verzia modulu
       prev_ver   : z ktorej verzie vznikla
       who        : kto vvytvoril tento zaznam (iba 1 z 5 moznosti
, t.j. je zodpovedny sa spravne udaje v nom
  -->
  <!ATTLIST record
```

```
            when        CDATA   #REQUIRED
            curr_ver    CDATA   #REQUIRED
            prev_ver    CDATA   #REQUIRED
            who         (dalxo | ico | chrco | hollub | kexo)
#REQUIRED
  >

  <!--
       short_desc : kratky popis zmeny. Ak nestaci, muoze sa viac
popisat v tele elementu (vid priklad).
                    Kratky popis je naschval povinny, pretoze musi
byt jasne a strucne povedane, co sa zmenilo (pri rychlom
vyhladavani).
                    Ostatne ako napr. implementacne deatily a
vykecavacky puojdu do toho tela elementu feature (ak treba).

  -->
  <!ATTLIST feature
            desc  CDATA   #REQUIRED
  >


  <!--
       id           : referencia na subor (vid nizsie element file
a priklad)
       current_ver  : sucasna verzia suboru
       prev_ver     : predchadzajuca verzia (mozno nebude treba a
je zbytocny)
                      Tieto informacie o verzii sa tu uvadzaju
preto (aj ked nejake info o verzii suborov su v metainfo), lebo
kazdy subor
                      ma vlastny proces vyvoja a zmena v jeho
vnutri nemusi ovplynit cely modul. Napr. ak sa prerobi nejaka
                      funkcia, aby fungovala rychlejsie, alebo sa
niektore funkcie v subore viac modularizuju, teda vznikne viacej
mensich funkcii.

        Takto budeme vediet s akou verziou suboru sa pracuje (ak
sa zmeni).
        Budeme mat aj jasno, v zodpovednosti jednotlivych suborov
na novej featuriji.
  -->
  <!ATTLIST affected_file
            id            IDREF   #REQUIRED
            current_ver   CDATA   #REQUIRED
            prev_ver      CDATA   #REQUIRED
  >

  <!--
        Tento element je dobry na to, aby sme mali prehlad aj o
tych suborov, ktore sa nemodifikovali (teda neboli vo feature)
        ale chceme vediet verzie suborov tvoria tuto verziu. Ako
som spominal, subory sa muozu aj vnutorne menit bez toho, aby
        to malo za nasledok zmenu verzie modulu. Toto je dobre na
to, aby ked sa spravi vnutorna zmena subora a urobila sa v tom
```

chyba a teraz zistime, ze nefunguje ani modul, aby sme hned vedeli, ake verzie suborov sme pouzili.

Samozrejme, ak sme zmenili vsetky subory co su ako file element potom not_affected_file sa neuvadza (vid priklad nizsie).

```
       id           : referencia na subor (vid nizsie element file
a priklad)
       current_ver  : pouzita verzia suboru
       prev_ver     : predchadzajuca verzia (mozno nebude treba a
je zbytocny)

  -->
  <!ATTLIST not_affected_file
            id            IDREF   #REQUIRED
            current_ver   CDATA   #REQUIRED
            prev_ver      CDATA   #REQUIRED
  >

  <!--

      id        : identifikator, v ramci XML musi mat jedinecny
nazov (v ramcy vsetkych atributov typu ID). XML nevie semanticky
                  rozlisit napr. ci vo vasom atribute IDREF alebo
IDREFS sa miesto id elementu auto odkazujete na od elementu
person.
                  XML vielen zabezpecit, ze ID bude jedinecne v
celom XML subore.

      name      : cesta a nazov suboru
      since     : odkedy je pridany do modulu
      since_ver : od ktorej verzie bol pridany do modulu
      short     : strucny, alebo kratky popis suboru (nepovinne).
                  Ak chcete popisat k suboru viac, takto muozete v
ramci ementu file -vid. example.

      Poviete si, ze naco je atribut id a aj name, ked sa rovnaju.
To nemusi byt zase pravda uplne. Preco?
      Lebo:
        Chcem, aby bolo jasne, dole v tych zaznamoch pri zmene
verzie, ktore subory to mali na svedomi.
        XML-ko podporuje identifikatory a referencie na ne, lenze
tie identifikatory a referencie (teda ID a odkaz na nu IDREF)
        musia byt "XML name".
        "XML name" znamena, ze lubovolne pismeno, cislica a len
tieto znaky: -, _ a . su povolene.
        Ako vidite, tak tam nie je \ a ani /. Preto v atribute
name bude cesta a meno suboru, teda nazov file a v id atribute
bude
        "prijatelna" forma mena, aby sa dalo v ramci XML
referencovat.
  -->
  <!ATTLIST file
            id          ID      #REQUIRED
            name        CDATA   #REQUIRED
```

```
            since       CDATA   #REQUIRED
            since_ver   CDATA   #REQUIRED
            short       CDATA   #IMPLIED
>
```

Príloha B - Zápisnice

Nachádzajú sa tu zápisnice z regulérnych stretnutí tímu. Vizuálnu stránku jednotlivých zápisníc upravoval Dalimír Orfánus.