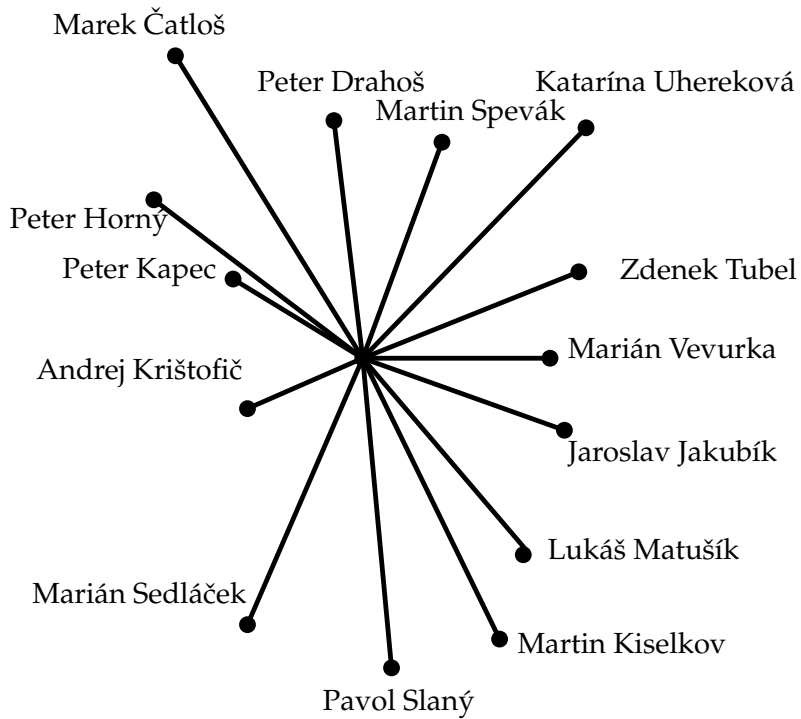


Web inžinierstvo



Lektor: doc. Ing. Mária Bieliková, PhD.

Obsah

Úvod	vii
1 Procesy a web inžinierstvo	1
1.1 Prečo potrebujeme proces vo web inžinierstve?	1
1.2 Prečo AWE proces?	4
1.3 Životný cyklus AWE procesu	7
1.3.1 Biznis analýza	8
1.3.2 Požiadavky	9
1.3.3 Návrh	10
1.3.4 Implementácia	10
1.3.5 Testovanie	10
1.3.6 Vyhodnotenie	12
1.3.7 Iteratívny a inkrementálny vývoj	13
1.3.8 Začiatok AWE procesu	15
1.4 Vyhodnocovanie a techniky na mieru	16
1.5 WebE proces - Proces web inžinierstva	18
1.5.1 Dlhší procesný cyklus	18
1.5.2 Kratší procesný cyklus	21
1.6 Hypertextový procesný model	22
1.7 Zhrnutie	23
2 Kategórie web aplikácií	25
2.1 Informačné web aplikácie	26
2.2 Download web aplikácie	26
2.3 Interaktívne web aplikácie	27
2.4 Vyhľadávacie web aplikácie	29
2.5 Transakčne orientované web aplikácie	30
2.6 Web aplikácie poskytujúce služby	31
2.7 Aplikácie s prístupom k databáze	32

2.8	Dátové sklady	33
2.9	Voľne dostupné služby	34
2.9.1	Webmail aplikácie	34
2.9.2	Webhosting	34
2.10	On-line hry	35
3	Návrh web aplikácií	39
3.1	Hypermédiá	39
3.2	Modely a metódy návrhu web aplikácií	40
3.2.1	Hypertext Design Model (HDM)	42
3.2.2	Object Oriented Hypermedia Design Methodology (OOHDM)	43
3.2.3	The Relationship Management Methodology (RMM)	44
3.3	Návrhové vzory	45
4	Návrh rozhraní	49
4.1	Základy používateľského rozhrania	49
4.1.1	Jasné navigačné pravidlá	50
4.1.2	Žiadne konečné stránky	50
4.1.3	Efektívny prístup	50
4.1.4	Dĺžka odozvy a interakcia	50
4.1.5	Jednoduchosť a konzistencia	51
4.1.6	Integrita a stabilita návrhu	51
4.1.7	Spätná odozva a dialóg	51
4.2	Návrh štruktúry web sídla	51
4.2.1	Organizovanie publikovaných informácií	52
4.2.2	Základné štruktúry stránok	52
4.3	Typy web stránok	53
4.3.1	Hlavné stránky	53
4.3.2	Menu a podstránky	55
4.3.3	Príbuzné stránky	55
4.3.4	FAQ stránky	55
4.3.5	Chybová stránka 404	55
4.4	Typografia	56
4.4.1	Čitateľnosť	56
4.4.2	Zarovnanie	57
4.4.3	Typy fontov	57
4.4.4	Čitateľnosť fontu na obrazovke	57
4.4.5	Fonty navrhnuté pre obrazovku počítača	57
4.4.6	Výber fontu	57
4.5	Redakčný štýl	58

4.5.1	Štylizovanie prózy	58
4.5.2	Nadpisy a podnadpisy	59
4.5.3	Odporúčané textové štýly	59
4.6	Záver	59
5	Testovanie a nasadenie	61
5.1	Prečo testovať	61
5.2	Testovanie vo všeobecnosti	62
5.3	Rozdiely medzi testovaním WA a klasických SA	63
5.4	Návrhy testovania web aplikácií	65
5.4.1	Dynamické testovanie metódou čiernej skrinky	65
5.4.2	Analýza a reprodukcia chýb vo web aplikáciách	66
5.4.3	Testovanie bezpečnosti vo web aplikáciách	67
5.5	Techniky, metódy, podporné programy a systémy	68
5.6	Nasadenie web aplikácií	68
6	Manažment	71
6.1	Motivácia	71
6.2	Plánovanie	73
6.2.1	Plán projektu	73
6.3	Roly v tímoch	76
6.3.1	Zákazník	77
6.3.2	Projektový manažér	77
6.3.3	Producent	78
6.3.4	Redaktor	79
6.3.5	Informačný architekt	79
6.3.6	Grafický dizajnér	79
6.3.7	Tvorca stránok	79
6.3.8	Hlavný programátor	80
6.3.9	Programátor	80
6.3.10	Databázový expert	80
6.3.11	Manažér kvality	81
6.4	Manažment rozsahu	81
6.4.1	Základný trojuholník - Čas, Náklady, Požiadavky na aplikáciu	81
6.4.2	Požiadavky na zmenu	82
6.4.3	Správa dokumentov	82
6.4.4	Iteratívne prístupy	82
6.4.5	Zhrnutie	83

7	Technológie a štandardy	85
7.1	Klient	85
7.1.1	Internet	85
7.1.2	Hypertext	86
7.1.3	Značkovacie jazyky	86
7.1.4	HTML	87
7.1.5	Multimédiá	90
7.2	Server	92
7.2.1	DNS (Domain Name System)	92
7.2.2	Elektronická pošta	93
7.2.3	HTTP	96
7.2.4	Web server	97
7.2.5	Proxy server	99
7.2.6	SQL server	99
7.2.7	Bezpečnosť na strane servera	100
	Záver	103
	Literatúra	107

Úvod

Odhady úsilia potrebného na implementáciu web aplikácie sa veľmi rôznia. Na jednej strane sa často stretávame s tvrdením, že web stránky možno tvoriť i s nezaškoleným personálom, pomocou jednoduchých nástrojov. Na strane druhej sú spoločnosti, ktoré utrácajú milióny za implementáciu a údržbu sofistikovaných a komplexných web aplikácií.

K vývoju mohutných a drahých aplikácií je nevyhnutné pristupovať systematicky a organizovane. Web aplikácie kombinujú aspekty vývoja softvéru, dokumentov a často i typografie či grafického dizajnu. Charakteristickými črtami dnešných web aplikácií sú predovšetkým:

- rýchlo sa meniace, heterogénne technické prostredie
- rýchlo a často sa meniaci obsah i zameranie
- distribuovaná architektúra (klientské prehliadače a HTTP servery)
- vysoký dôraz na štruktúru informácií a hypertextovú navigáciu
- štruktúra hypertextu odráža sémantiku jeho obsahu
- obsluha a práca viacerých používateľov súčasne
- nepretržitá prevádzka a údržba počas prevádzky

Softvérové inžinierstvo sa venuje problematike návrhu, tvorby a riadenia vývoja informačných systémov. Často sa možno stretnúť s pohľadom na softvérové inžinierstvo ako na pavelu, ktorá nemá presne definované pravidlá. Z pohľadu vied ako matematika, fyzika alebo chémia je to určite tak. Treba si ale uvedomiť, že na rozdiel od pevných základov matematiky alebo chémie (ktoré sú v princípe nemenné), prostredie, v ktorom sa softvér tvorí a vyvíja, sa neustále mení. Ďalším argumentom je i skutočnosť, že softvérové inžinierstvo dosahuje sotva 20 rokov svojej existencie a už vykazuje známky členenia na rôzne disciplíny.

Významnou disciplínou softvérového inžinierstva je aj web inžinierstvo. Ide o disciplínu, ktorá sa zaoberá tvorbou web aplikácií. Web aplikácie sa líšia od klasického softvéru v mnohých ohľadoch. Jedným z najväčších rozdielov je najmä ich krátka doba vývoja a ďalší vývoj počas prevádzky. Častokrát sa požiadavky na web aplikácie menia i počas ich tvorby, čo vyžaduje použitie sofistikovaných metód návrhu a riadenia projektov. Navyše, web aplikácie sú závislé na obrovskom množstve štandardov a technológií, ktoré sa často menia, vznikajú a zanikajú. Charakter web aplikácií nás núti klásť dôraz na kompatibilitu a testovanie. Ide zväčša o distribuované viacpoužívateľské systémy, ktorých používatelia využívajú celú plejádu rozdielneho hardvéru, ako napríklad mobilné telefóny, PC či iné zariadenia. Tieto zariadenia navyše majú častokrát odlišný softvér a operačný systém.

Kniha sa bližšie zaoberá, ako jej názov napovedá, web inžinierstvom. Predpokladáme, že čitateľ je oboznámený so základnými postupmi tvorby klasického softvéru, ako je napríklad vývoj na základe vodopádového modelu.

Najväčšie rozdiely medzi tvorbou klasického softvéru a web aplikácií možno pozorovať v procese jeho tvorby, ktorý je opísaný v prvej kapitole. Kapitola sa venuje najmä AWE procesu a bližšie opisuje jeho výhody a dôsledky na riadenie projektu. V ďalšej kapitole nasleduje stručný prehľad web aplikácií, ktorý sa snaží kategorizovať aplikácie podľa poľa ich pôsobnosti, použitých techník a technológií. Nasledujú kapitoly venované návrhu web aplikácií a ich používateľských rozhraní. Tieto predstavujú úvod do návrhu používateľských rozhraní, grafického návrhu ako aj do návrhu architektúry web aplikácií samotných.

Neoddeliteľnou súčasťou každého projektu je testovanie. Venuje sa mu kapitola o testovaní, ktorá poukazuje na problémy testovania multipoužívateľských aplikácií v distribuovanom prostredí Internetu. Nasleduje kapitola, ktorá sa zaoberá špecifikami plánovania a riadenia web projektov. Záver knihy je venovaný prehľadu technológií a protokolov. Vysvetľuje ich použitie a význam pre web aplikácie.

Zoznam skratiek

ASP	Active Server Pages
AVI	Audio Video Interleave
AWE	Agile Web Engineering
BIFS	BIinary Format for Scenes
CSS	Cascading Style Sheet
DCT	Discrete Cosinus Transformation
DNS	Domain Name System
DTD	Document Type Definition
FC	Request For Comments
FTP	File Transfer Protocol
GML	Generalized Markup Language
GUI	Graphic User Interface
HDM	Hypertext Design Model
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IT	Informačné technológie
JDBC	Java Database Connectivity
JPEG	Joint Photographics Experts Group
JSP	Java Server Pages
LZW	Lempel-Ziv-Welch
MPEG	Motion Picture Experts Group
ODBC	Open Database Connectivity
OLAP	Online Analytical Processing
OOHDM	Object-Oriented Hypermedia Design Methodology

POP3	Post Office Protocol v3
RAD	Rapid Application Development
RDBMS	Relational Database Management System
RMDM	Relationship Management Data Model
RMM	Relationship Management Methodology
SGML	Standard Generalized Markup Language
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSH	Secure SHell
TCP	Transmission Control Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VBS	Visual Basic Script
VRML	Virtual Reality Markup Language
WAE	Web Application Extension
WBS	Work Breakdown Structure
WWW	World Wide Web
XML	eXtensible Markup Language
XP	eXtreme Programming

Kapitola 1

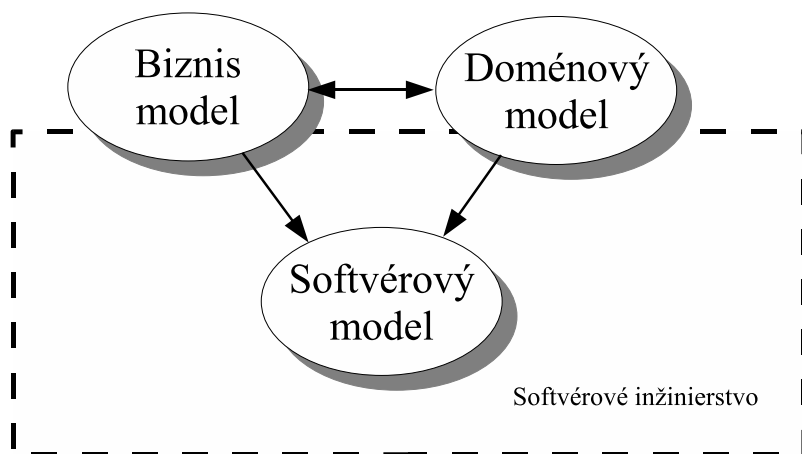
Procesy a web inžinierstvo

1.1 Prečo potrebujeme proces vo web inžinierstve?

Tradičné projekty softvérového inžinierstva sú zamerané hlavne na tvorbu softvérových súčastok pre podporu systémov, ktoré sú často generické. Tieto softvérové súčastky a podporné systémy sa často vyvíjajú nezávisle od údajov, ktoré používajú. Na druhej strane web inžinierstvo sa zameriava na tvorbu softvérových súčastok a podporných systémov, ktoré sú vyvíjané paralelne s tvorbou údajov. Z toho vyplýva, že každý web projekt je riešením na objednávku, ktoré zahŕňa softvér, ale i údaje. Nie je možné dodávať riešenia bez naplnených údajov, a aj preto je pre úspech projektu potrebné zvýšiť úsilie pre porozumenie zákazníka a všetkých jeho požiadaviek. Je potrebné si uvedomiť, že web aplikácie sú spravidla založené na ponuke prístupu k informáciám. Týchto prístupov je pomerne veľké množstvo a netreba zabúdať na nutnosť ich dôkladného zváženia a následného výberu. Najdôležitejšie pre úspech web aplikácie je zamerať sa na porozumenie požiadaviek koncového používateľa, pre ktorého je systém navrhovaný.

V softvérovom inžinierstve rozoznávame tri výrazné modely, ktoré je potrebné brať do úvahy, a ktoré zodpovedajú ich tradičným procesom. Sú to: softvérový model, doménový model a biznis model.

Na obrázku 1.1 znázorňujeme vzájomné ovplyvňovanie sa jednotlivých modelov v tradičnom procese softvérového inžinierstva. Softvérový model je ovplyvnený iba malou časťou doménového a biznis modelu. Ovplyvňovanie sa uskutočňuje iba raz alebo dvakrát počas vývojového životného cyklu. Informácie, ktoré softvérový model získa vzhľadom na biznis a doménový model, sú v prvom rade brané ako statické. Zriedkakedy má softvérový model výrazný vplyv na biznis a doménový model. Dôsledkom je, že veľa existujúcich softvérových riešení, ktoré vychádzajú z tradičných procesov softvérového inžinierstva, sú veľké implementácie existujúcich



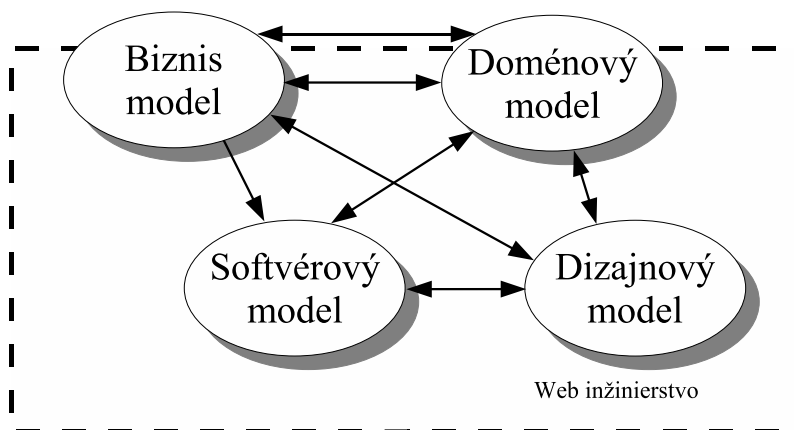
Obrázok 1.1: Interakcia medzi modelmi zodpovedajúcim tradičným procesom v softvérovom inžinierstve [MW01]

firemných praktík.

Vo web inžinierstve pridávame model vychádzajúci z potreby estetického používateľského prostredia a nazývame ho dizajnový model. Web inžinierstvo má väčší dosah na biznis a doménový model, s ohľadom na návrh a vývoj údajov ovplyvňujúcich web stránky. Preto neovplyvňuje a nemení len softvérový a dizajnový model, ale potrebuje mať dosah aj na biznis a doménový model a v prípade potreby mať možnosť ich zmeniť. Vo web inžinierstve sa biznis a doménový model musia navzájom ovplyvňovať a prispôsobovať. Toto vzájomné ovplyvňovanie je znázornené na obrázku 1.2.

Pri vývoji web projektu vzniká často potreba robiť určitý stupeň reinžinierstva¹ v uvedených modeloch. Takéto vzájomné ovplyvňovanie sa spomínaných štyroch modelov sa niekedy stretáva s nepochopením u zákazníka. Každá organizácia povolí reštrukturalizáciu svojej firmy iba v prípade očakávania zisku. Preto reinžinierska iniciatíva potrebuje adaptovať dôležité modely iba tak, aby zaistili pochopenie u zákazníka. Je veľmi dôležité, aby sa organizácia a zainteresovaní ľudia vo web projekte snažili pochopiť tieto vplyvy medzi modelmi vo web inžinierstve. Napokon si treba uvedomiť, že organizácia a aj jednotlivci, ktorí nepochopia význam reinžinierstva počas vývoja web aplikácie, riskujú úspech svojho projektu a

¹Reinžinierstvo = spätné inžinierstvo + reštrukturalizácia. Spätné inžinierstvo je proces analýzy systému s cieľom identifikovať súčiastky systému a vzťahov medzi nimi a zároveň vytvoriť reprezentáciu systému v inej forme alebo na vyššej úrovni. Reštrukturalizácia je transformácia systému z jednej reprezentácie do inej, na rovnakej úrovni abstrakcie.



Obrázok 1.2: Interakcia medzi modelmi zodpovedajúcim procesom vo WE [MW01]

tiež aj dlhodobé prežitie svojej organizácie.

Pre lepšie pochopenie predchádzajúcich modelov si uvedieme príklad. Predstavme si veľkosklad, ktorý chce zaviesť elektronické objednávky a obchod. Je jasné, že každý veľkosklad (nie len ten náš jeden) musí mať pokladňu, odberateľov, dodávateľov, nejaký príjem, výdaj atď. Toto všetko zahŕňame do doménového modelu. Dopad vytvorenia elektronického obchodovania je, že tento model sa musí rozšíriť o distribútorov, ktorí prijímajú elektronické objednávky, a o ďalšie technológie, počítače atď. Biznis model súvisí iba s našim veľkoskladom, určuje ho štruktúra firmy. Uvádza, ktorá funkcia vo firme pripadá ku ktorej osobe, čo má na starosti a aké má povinnosti. Tiež obsahuje počet pokladní vo firme. Po nasadení web aplikácie sa musí urobiť reštrukturalizácia tohto modelu (reštrukturalizácia firmy), čiže sa zmenia úlohy niektorých zamestnancov pridelením k obsluhu nového elektronického systému. Softvérový model obsahuje návrh požadovaného systému a jeho implementáciu. Dizajnový model zahŕňa návrhy obrazoviek, používateľské rozhranie atď.

Týmto nechceme naznačiť, že elementy biznis a doménového modelu môžeme vyvíjať nezávisle od seba, ale práve naopak. Biznis a doménový model majú vo web inžinierstve podstatnú a kritickú účasť na úspechu web aplikácie. Preto oba modely treba vyvíjať z rôznych pohľadov softvérového modelu. Je nereálne, aby jeden individuálny člen tímu porozumel všetkým pohľadom vyplývajúcich z rôznych ovplyvňujúcich modelov vo web inžinierstve. Takže je dôležité, aby vývojový tím zahŕňal firemného experta, doménového experta, manažéra, dizajnéra aj softvérového inžiniera. Zároveň by sa každý z nich mal vzdelávať vo svojej odbornej

oblasti a spolupracovať s ostatnými v tíme, v záujme dosiahnutia čo najlepšieho riešenia, alebo zachytenia projektových a organizačných problémov. Znamená to, že členovia tímu potrebujú pochopiť dopad ovplyvnenia sa ich modelov a preniesť získané porozumenie do procesu reinžinierstva. Firemný proces reinžinierstva je téma sama o sebe, ale AWE (Agile Web Engineering) proces sa pokúša zachytiť tento mechanizmus (reinžinierstva) vo vývoji web aplikácií. Je dôležité uvedomiť si dôležitosť firemného procesu v reinžinierstve a zabezpečiť komunikačný mechanizmus a štruktúru vývoja web aplikácie tak, aby podporili reinžinierske aktivity.

Projekt je komplikovaný aj samotným softvérovým modelom a jeho dopadom na biznis a doménový model. Zavedenie web technológií môže mať dopad na niektoré terajšie operačné metódy, ktoré sa môžu stať nadbytočnými, ale na druhej strane môžu poskytnúť i nové firemné príležitosti. Dopad spôsobený web technológiami bude mať priamy vplyv na biznis a doménový model. Dizajnový model (novo pridaný) je čiastočne zodpovedný za otázky súvisiace s návrhom používateľského rozhrania a zameraním sa na estetický aspekt web aplikácie. Biznis a doménový model budú tiež pôsobiť na dizajnový model. Konflikt často vznikne medzi estetickým dizajnom a použiteľnosťou. Vzťah medzi softvérovým modelom a dizajnovým modelom má teda dve cesty dopadu.

Ak sa dopad medzi softvérovým, dizajnovým, doménovým a biznis modelom týka problémov vzťahujúcich sa na vývoj web aplikácií, tak sa má daný dopad odrážať v procese web inžinierstva. Je podstatné, aby si vývojový tím počas vývoja bol vedomý existencie rôznych vzťahov medzi modelmi. To môže celý tím využiť nielen pri dodaní systému, ale tiež pri reinžinierskych aktivitách vo firemnom procese. Momentálne ovplyvňovanie sa modelov v softvérovom procese nezodpovedá potrebám web inžinierstva.

V ďalšej časti opíšeme agilný proces a objasníme, prečo táto cesta poskytuje dobré adresovanie úloh v procese web inžinierstva. Následne v krátkosti opíšeme WebE procesný model a hypertextový procesný model.

1.2 Prečo AWE proces?

Pojem agilný sa zaviedol na začlenenie množstva moderných prístupov pri tvorbe softvérových produktov. Patrí sem extrémne programovanie (angl. Extreme Programming), adaptívny softvérový vývoj (angl. Adaptive Software Development) a metodika dynamického vývoja systémov (angl. Dynamic Systems Development Methodology). Sedemnást' zástancov a metodológov zo skôr spomenutých a iných agilných procesov sa stretla vo februári 2001. Výsledkom tohto stretnutia bolo vytvorenie tzv. „Agile Alliance“ a vznik dokumentu „The Manifesto for Agile Software Development“.

Nasledujúci text vyššie spomenutého dokumentu dáva dobrý prehľad o cieľoch a zámeroch Agile Alliance: Odhaľujeme lepšie spôsoby vývoja softvéru tým, že ho robíme a pomáhame iným, aby ho robili. Našou prácou sme dospeli k nasledovnému poznaniu:

- (i) jednotlivci, spolupráca > procesy, nástroje
- (ii) funkčný softvér > vyčerpávajúca dokumentácia
- (iii) spolupráca so zákazníkom > podmienky zmluvy
- (iv) reagovanie na zmeny > uskutočnenie plánu

Pozn.: Znamená to, že si ceníme veci na pravej strane, ale veci na ľavej si ceníme viac.

O úspechu vyvíjanej web aplikácie rozhodujú vývojári a organizácie zapojené do web projektu. Metodológia AWE alebo iný proces môžu byť nanajvýš druhoradým dôvodom úspechu projektu. Je teda jasné, že voľba agility so zreteľom na ľudí a bod (i) sa hodí pre vývoj web aplikácie.

Existuje nepísané pravidlo, že ľudia sú najdôležitejší a to je dôvodom, prečo nezačali vznikať monumentálne procesy spojené s vývojom web aplikácií. Veľa monumentálnych procesov sa pokúša kodifikovať dobré zvyky a skúsenosti príliš detailne a to iba kvôli vývojárom, ktorí nerozumejú dôležitosti ich práce. To vedie často k tomu, že monumentálne procesy sa používajú ako kuchárske knihy, kde sú vývojári vtiahnutí do falošného pocitu istoty tým, že slepo nasledujú recept namiesto toho, aby používali suroviny jednotlivo. Tak by totiž vytvorili softvér, ktorý by bol riešením ich problémovej oblasti.

Koniec koncov, vývojové tímy používajú monumentálne procesy zriedkavejšie ako plánovali. Projekty s takýmto prístupom často venujú väčšinu svojho úsilia produkovaniu veľkého množstva dokumentácie namiesto toho, aby sa zamerali na dodanie funkčného softvéru. To by malo byť hlavným predmetom ich činnosti.

AWE procesy tlačia na vývojárov a na organizácie zapojené do vývoja, aby splnili hlavné ciele projektu. To je prvý a hlavný cieľ AWE. To neznamená, že by sme dokumentáciu a metodológiu nepovažovali za dôležitú. Naopak! Tie hrajú dôležitú úlohu na ceste k úspechu, len veríme, že najdôležitejším produktom web aplikácie je ona sama, tak ako je uvedené v bode (ii).

Uvažujme situáciu z 1.2. Veríme, že intenzita zmien v rámci týchto modelov a dopad zmien v jednej časti na ostatné je zahrnuté v AWE procese bodmi (iii) a (iv). K už spomenutému cieľu „The Manifesto for Agile Software Development“, uvedieme navyše aj 12 princípov, ktoré obsahuje a ktorých sa zastávame:

- (A) Našou najvyššou prioritou je uspokojiť zákazníka skorým a nepretržitým prísunom softvéru.

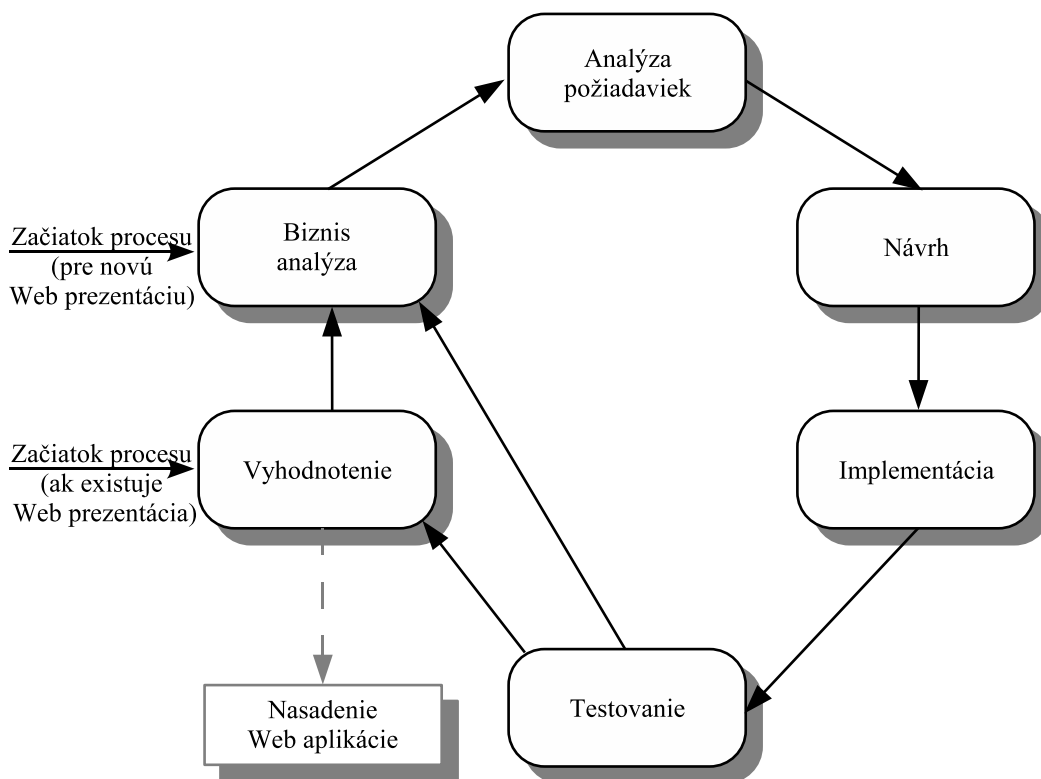
- (B) Prijímať zmeny požiadaviek aj v neskorých etapách vývoja.
- (C) Doručovať funkčný softvér často v intervale od pár týždňov do pár mesiacov, s preferenciou kratších období.
- (D) Ľudia z firmy aj vývojári pracujú spolu denne na projekte.
- (E) Projekt stavajte na motivovaných individualitách. Poskytnite im prostredie, splňte ich požiadavky, verte, že projekt a prácu na ňom zvládnu.
- (F) Najproduktívnejším a najefektívnejším spôsobom odovzdávania informácií je priama konverzácia tvárou v tvár (angl.: face-to-face).
- (G) Funkčný softvér je primárnym merítkom pokroku.
- (H) Agilný proces podporuje udržateľný vývoj. Sponzori, vývojári a používatelia by mali spolu držať konštantný krok.
- (I) Pretrvávajúci záujem o perfektnosť postupov a dobrý návrh zvyšuje agilitu.
- (J) Jednoduchosť (maximalizovaním množstva práce, ktorá sa nemusí urobiť) je hlavná a dôležitá vec.
- (K) Najlepšie architektúry, požiadavky a návrh sa vytvoria v samovznikajúcich tímoch.
- (L) Je dôležité, aby členovia tímu v pravidelných intervaloch premýšľali, ako by sa ich práca dala zefektívniť a podľa toho sa aj zriadili.

AWE je iteratívny a inkrementálny proces - **(A)**, **(B)**, **(C)**. AWE proces podporuje vývoj v rôznych disciplínach. AWE vedie vývojárov k tomu, aby pracovali každodenne v tesnej blízkosti **(D)**, v prostredí, ktoré podporuje plodný vývoj. Toto v spojení s bodom **(F)** robí vývoj ešte viac tvorivým. Považujeme za zodpovednosť vývojára a tímu (organizácie), aby priebežne doručovali web aplikácie, ktoré uspokojujú koncového používateľa, pretože to je jediným merítkom úspechu **(G)**.

Doručovanie funkčného softvéru by malo byť udržateľné počas celého AWE procesu konštantnou frekvenciou **(H)**. V zásade je na každom vývojárovi, aby udržoval AWE procesy v chode. Robí to integrovaním postupov a vytváraním jednoduchého **(J)**, kvalitného návrhu **(I)** a implementáciou web riešení (angl. web-based solutions). AWE dáva za zodpovednosť vývojáorskemu tímu vplyv na architektúru, požiadavky a samotný návrh vytváraných web aplikácií **(K)**. Pri každej iterácii životného cyklu sa vyžaduje od vývojového tímu, aby vyhodnotil vývojový proces **(L)**.

1.3 Životný cyklus AWE procesu

AWE proces identifikuje všetky hlavné etapy vývoja web aplikácie, ktoré by sa mali vykonať. Každý, kto už má nejaké skúsenosti so softvérovými procesmi, hlavne s vodopádovým modelom, sa pri pohľade na obrázok 1.3 musel otriast', pretože veľa zmien mu je povedomých. Ale tu našťastie končí akákoľvek príbuznosť. Na obrázku 1.3 je znázornený životný cyklus AWE procesu. Jediným výstupom, ktorý musí byť z AWE procesu, je web aplikácia sama. To ale neznamená, že nie je potrebné využívať výhody prechodných dokumentov, diagramov alebo iných poznámok². Zodpovednosť je na organizácii a vývojároch aby, ak je to potrebné, našli, integrovali, vyhodnotili a vytvorili metódy, ktoré by podporili aktivity na obrázku 1.3.



Obrázok 1.3: Životný cyklus AWE procesu [MW01]

²často je to vhodné a niekedy aj potrebné, ale nie nutné

Poslednou dobou si extrémne programovanie (XP) ako agilný proces získava veľkú pozornosť. V XP sa vývojárom odporúča, aby spolu komunikovali pomocou zdrojového kódu. Toto docielime programovaním vo dvojiciach alebo inými XP technikami. Na rozdiel od tvorby klasických softvérových systémov, súčasťou výstupov pri tvorbe web aplikácií sú aj údaje integrované v produkte. Web inžinierstvo sa zakladá na malých vývojárskych tímoch, ktoré sú z rôznych oblastí, a preto vo väčšine prípadov nebudú schopné komunikovať zdrojovými kódmi aplikácie. Mali by komunikovať na základe svojich skúseností s aplikáciou v prehliadači.

Zoberme si také elektronické múzeum a on-line obchod umeleckej galérie. Sú už hojne zastúpené na Internete a vyvíjajú sa viac ako 8 rokov so zámerom pomôcť pri vzdelávaní školopovinných detí, ktoré spomenuté miesta nemôžu navštíviť z dôvodu veľkej vzdialenosti. Hlavnými návštevníkmi týchto stránok sú deti, dospelí, ktorí sa zaujímajú o túto problematiku príležitostne, resp. akademici so záujmom o múzejné kolekcie. Pretože web riešenie má osloviť tak rozdielne publikum, vývojári musia vynakladať enormné úsilie, aby očakávania všetkých skupín boli čo najviac uspokojené. Jedným spôsobom ako to doceliť je neustále sledovať spätnú väzbu náhodne vybraných zástupcov týchto skupín.

1.3.1 Biznis analýza

Účelom etapy biznis analýzy je pochopiť problémy, ktorým sa má venovať žiadaná web aplikácia. To je v skutočnosti ťažšie ako sa zdá. Často je táto fáza realizovaná rôznymi skupinami vývojárov alebo zainteresovanými ľuďmi. Ak sa z týchto analýz má vytvoriť nejaké riešenie, tak je bezprostredne nutné plne pochopiť problematiku, čo je aj dôvodom prítomnosti zainteresovaných ľudí. Je nevyhnutné, aby bol prístup každého vývojára v tejto fáze otvorený, keďže je v záujme nás všetkých, aby sa tieto skutočné problémy našli.

Dnes sa vyvíja veľa web stránok len preto, že zainteresované organizácie nechcú zaostávať za konkurenciou. Už sa ale nezameriavajú na to, aby aj korešpondovala s ich firemnými aktivitami.

Je treba veľa spoločného úsilia a veľkého množstva krokov v životnom cykle, kým sa odhalia tie pravé témy, ktorým sa má venovať žiadaná web aplikácia. Každou iteráciou v životnom cykle sa rozširuje problémová oblasť, ktorej sa treba venovať. Ak sa tým nájde najrizikovejšia úloha, tak by sa mala riešiť ako prvá. Tam sa skrýva riziko najväčších zmien a to nám umožňuje vytvoriť prototyp web aplikácie.

Väčšina projektov začne etapou AWE procesu Biznis analýza. Návrat do fázy Vyhodnotenie by bol múdry, ak tím usúdi, že opísať a zhodnúť sa na problémovej oblasti je veľmi ťažké. Dôvodom je, že tu sa vyhodnocujú už existujúce web prezentácie alebo stránky konkurencie, aby sa lepšie pochopili problémy. Taktiež by

sa v tejto fáze mala zodpovedať otázka: „Vytvorili sme dobrý produkt?“

Ešte raz zoberme do úvahy prvú iteráciu AWE procesu v našom príklade elektronického múzea a on-line obchodu umeleckej galérie. Počas etapy Biznis analýza sa dospelo k záveru, že chceme zvýšiť počet zobrazených položiek v on-line obchode, kde sa zobrazuje všetko predané v múzeu a v umeleckej galérii, napr. kolekcie obrázkov objednaných poštou. Tím sa rozhodol vyhodnotiť iné web riešenia on-line umeleckých galérií, či niekto niečo podobné robí a poučiť sa z nájdených stránok.

Prieskum v praxi ukázal, že väčšina vývojových tímov má problémy s pochopením toho, čo sú požiadavky a ako sa vysporiadať s ich priebežnými zmenami. Ak sa plánuje použitie AWE procesov, tak by všetci vývojári mali túto etapu v každej iterácii opúšťať s jasným chápaním problémov. V každej iterácii by mal byť identifikovaný najdôležitejší problém. Tím by mal navyše začať určovať kritériá, podľa ktorých bude v etape vyhodnotenie určovať, či bol problém vyriešený.

Každý vývojár zapojený do nejakej iterácie AWE procesu by mal vedieť kladne zodpovedať nasledujúcu otázku: „Rozumiem problémom, ktoré sa majú riešiť v tejto iterácii AWE procesu?“ Zároveň by tím mal vedieť kladne zodpovedať na otázku: „Rozumie a súhlasí každý člen s problémami, ktoré sa majú riešiť v tejto iterácii AWE procesu?“ Ak aspoň na jednu z otázok niekto odpovie „nie“, treba sa vrátiť do kroku Biznis analýza a klásť na ňu väčší dôraz.

1.3.2 Požiadavky

Je nevyhnutné, aby každý člen tímu súhlasil s úlohami, ktoré sa majú riešiť, skôr ako sa začnú určovať požiadavky. V tejto fáze sa majú určiť funkcionálne požiadavky (čo má navrhované riešenie robiť) a nefunkcionálne požiadavky (čo bude navrhované riešenie potrebovať). Vo vývoji web aplikácií sa vyskytuje veľké množstvo nefunkcionálnych požiadaviek, ktoré musia byť pre úspešné zvládnutie projektu dôkladne určené. Predovšetkým tie, ktoré sa týkajú architektúry, vrátane používateľského rozhrania a použiteľnosti.

AWE neodporúča žiadne metódy. To neznamená, že nenabádame k ich používaniu. Iba ich nijako zvlášť nevyzdvihujeme. K daným technikám existuje veľa zdrojov, kde sa dá o nich dozvedieť viac. [CL00, JBR99, JCJÖ92, Jef01, MMC01, SW98, WK99]

Potom čo sa identifikujú požiadavky, by sa tím mal zamerať na začatie tvorby plánu využitého vo fáze Testovanie. Výsledkom bude možnosť tímu odpovedať na otázku: „Vytvorili sme produkt správne?“ Je dôležité, aby sa celý tím zhodol na tom, ako zostrojiť test na zodpovedanie spomenutej otázky. Inak môže projekt skrachovať, alebo riešenie nebude spĺňať dohodnuté požiadavky.

1.3.3 Návrh

Návrh zahŕňa pochopenie, koordináciu a prediskutovanie všetkých hlavných otázok komplexnej web aplikácie, ktoré nie sú zatažené implementáciou. Keď sa AWE proces používa na veľkých web projektoch, vyžaduje si to, aby vývojári rovnakého typu z rôznych tímov medzi sebou komunikovali. Táto komunikácia slúži k znovupoužívaniu nielen výsledných produktov, ale tiež návrhu architektúry. Ak chceme, aby projekt nasledoval rýchlo, bez vážnych problémov, s prenosnosťou a znovupoužitím návrhu, potom sa musí venovať veľká pozornosť architektúre.

Ak berieme do úvahy všetky metódy, ktoré podporujú návrh web aplikácií, tak by sme mohli nájsť veľmi veľa príkladov, ale nijaké špeciálne neodporúčame. Je na zodpovednosti každého zapojeného vývojára uvedomiť si všetky problémy spojené s vývojom web aplikácií.

1.3.4 Implementácia

Implementáciu považujeme za rovnako dôležitú ako návrh. Aj keď je veľmi veľa rozdielností v etape návrhu a implementácie, obe zahŕňajú rozhodnutia, ktoré vážne ovplyvnia úspech projektu. Agilný proces XP odporúča vývojárom v tímoch, väčšinou softvérovým inžinierom, aby komunikovali pomocou zdrojových kódov, čo sa bežne dosahuje programovaním v pároch alebo rozsiahlym testovaním. Na nešťastie sa tento spôsob komunikácie dá použiť len z pohľadu ľudí zaoberajúcimi sa softvérovým a dizajnovým modelom, ale nie je praktický pre všetkých vývojárov zo zmiešaného tímu. Odporúča sa komunikovať cez web rozhrania. Všetci vývojári by mali spolupracovať a sústrediť ich vývojárske úsilie na web rozhranie s využitím nejakých spoločných diskusií, kde s nimi získajú skúsenosti.

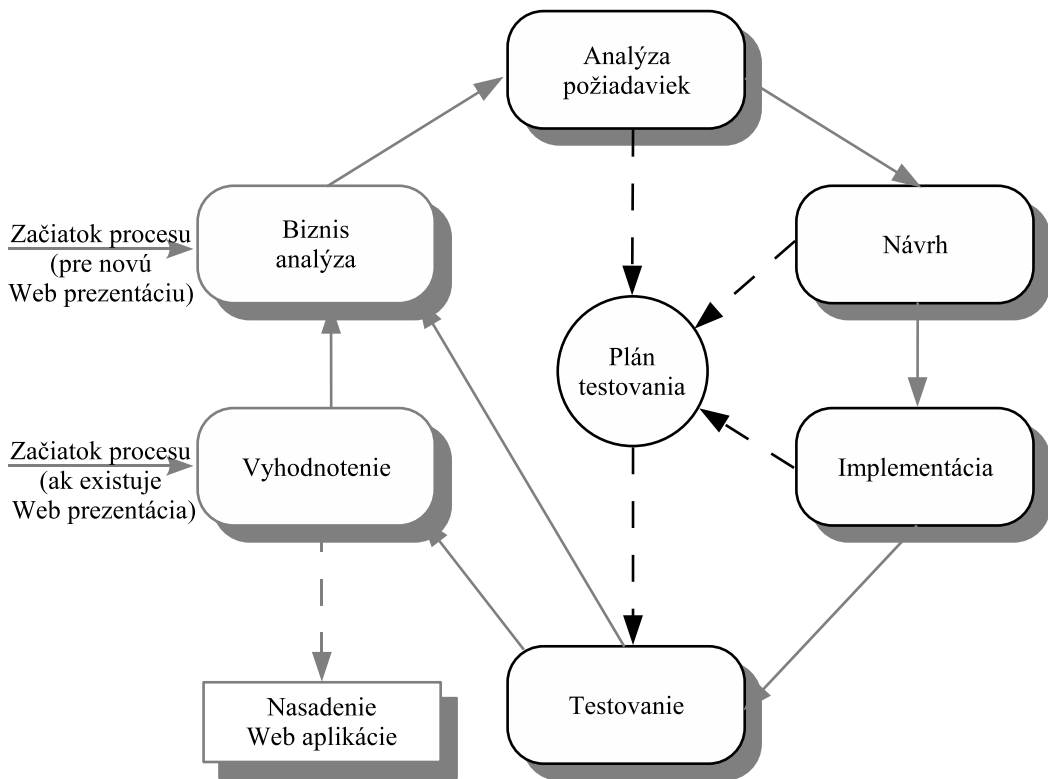
1.3.5 Testovanie

Testovanie je rozhodujúcou etapou v živote softvéru. Je podstatné aby si boli všetci vývojári vedomí účelnosti tejto etapy. Testovanie objektívne určí, čo bolo alebo nebolo vyvinuté uspokojivo podľa požiadaviek klienta. Všetci vývojári by si mali položiť otázku: „Ako vytvoriť správny produkt?“

Okrem testovania funkcionálnych požiadaviek je tiež veľmi podstatné, aby si všetci vývojári uvedomili dôležitosť testovania požiadaviek nefunkcionálnych. Vývojársky tím, by si mal minimálne položiť nasledujúce otázky počas testovania nefunkcionálnych požiadaviek: „Je naša aplikácia kompatibilná s prehliadačom, pre ktorý je určená a aké je obmedzenie nášho produktu (rozlíšenie, rýchlosť sieťového pripojenia, rýchlosť procesora, pamäť atď.)?“, „Vydrží naša aplikácia očakávané zataženie?“, „Je náš systém ľahko rozšíriteľný?“, „Je náš systém prenosný?“, „Zod-

povedá náš systém požadovaným používateľským obmedzeniam?“, „Je náš systém bezpečný?“

Testovanie potrebuje vstupy z etáp Analýza požiadaviek, Návrh a Implementácia. Počas etapy Analýza požiadaviek sa vykonáva testovanie na vyššej úrovni ($X \rightarrow Y$, systém bude zvládať 400 aktívnych používateľov v rovnakom čase) a počas etáp Návrh a Implementácia sa vykonáva testovanie na nižšej úrovni (Ak je $Z > 10$ počas transakcie A, tak systém bude radiť T počas R sekúnd). Pozri opis etapy na obrázku 1.4 zahrňujúci vytvorenie testovacieho plánu.



Obrázok 1.4: Vytvorenie testovacieho plánu v životnom cykle AWE procesu [MW01]

Veľa web vývojárov testuje už počas návrhu a implementácie. Toto môže tím často dostať do falošnej predstavy o správnosti vyvíjaného systému. Vývojári si musia byť vedomí, že dobré vlastnosti web rozhrania na ich počítači nemusia zaručiť dobré vlastnosti web rozhrania na počítači, pre ktorý je aplikácia určená. Aj

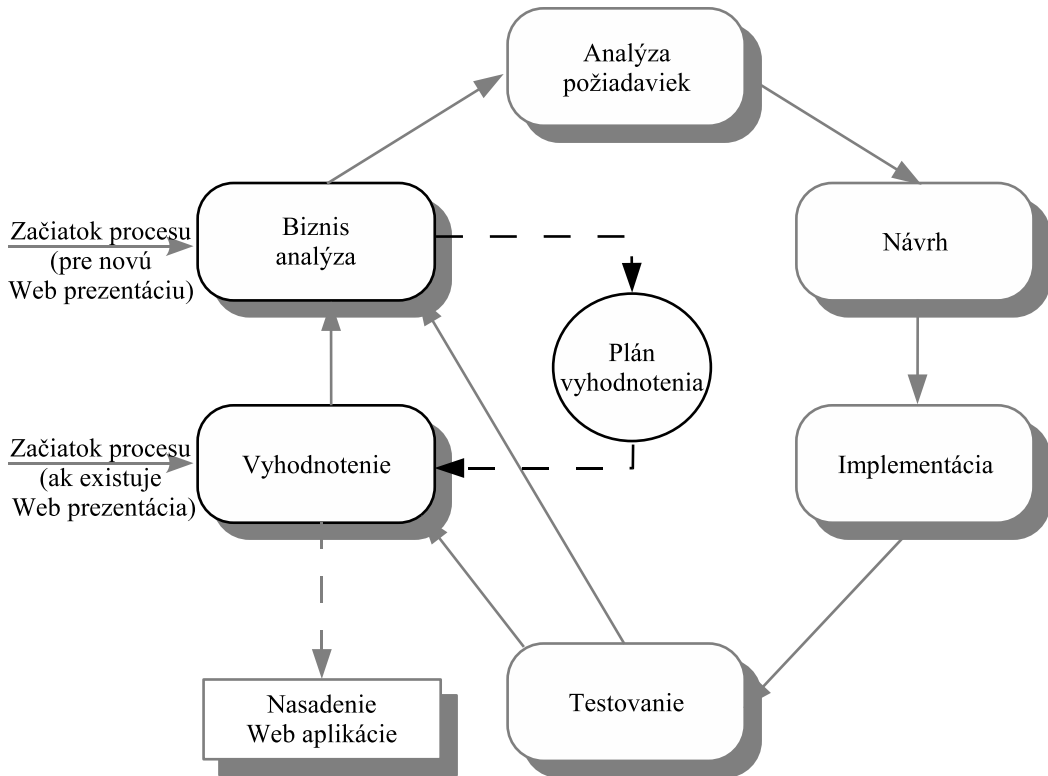
keď vývojári hĺbkovo testujú počas návrhu a implementácie, je základom, aby celý tím zahrnul testovanie aplikácie v prostredí, ktoré je tak blízke konečnému, ako je to len možné. Je dobré zabezpečiť, aby vývojársky tím mal prístup k minimálnej konfigurácii systému (jednoduchý prehliadač, šírka pásma na pripojenie, grafické rozlíšenie) pre testovanie.

1.3.6 Vyhodnotenie

Plán vyhodnotenia môže byť odvodený z otázok alebo úloh identifikovaných v etape Biznis analýza. Obrázok 1.5 opisuje etapu obsiahnutú v tvorbe plánu vyhodnotenia. Základná otázka, ktorú musí tím sledovať v etape vyhodnotenia pýtaním sa samých seba, je: „Zostavili sme správny produkt?“ Je to nutné, aby tím objektívne vyhodnotil výsledok nezávisle od výsledku návrhu a implementácie. Iba potom bude schopný každý oceniť či je, alebo nie je projekt riešením stanoveného problému. Ak je vyhodnotenie vykonané kvalifikovaným vývojárskym meraním s účasťou koncového používateľa, často vedie k lepšiemu porozumeniu problému, alebo vyriešeniu otázok. Toto nové porozumenie problémovej oblasti vyžaduje spätnú väzbu v etapách Biznis analýza a Požiadavky.

Je nutné, aby na vyhodnotenie výsledku boli použité reprezentatívne vzorky konečných používateľov. Pokiaľ je účasť koncového používateľa cenným zdrojom, nie je tu žiadna lepšia náhrada. Je tiež dôležité, aby vývojári porozumeli ako komunikovať s koncovými používateľmi počas vyhodnotenia. Aby sa ich pýtali, čo si myslia a aké nedostatky má aplikácia. Je nutné pozorovať použitie výsledného produktu koncovým používateľom v normálnom používateľskom prostredí.

Niektorí vývojári sa snažia vyhnúť etape Vyhodnotenie vo vývoji web aplikácií a berú ju ako prekážku pre doručenie ich projektu. Napokon sa táto etapa redukuje podľa toho, či si tím želá vytvoriť web produkt alebo chce vyriešiť problém s web aplikáciou. Vyhodnotenie môže spotrebovať veľa času a tím je aj drahé. To ale nutne neznamená dlhší čas na vývoj a väčšie ohodnotenie. Ak chcete určiť otázky spojené s vývojom web aplikácie, tak vyhodnotenie vám ich pomôže skôr identifikovať a lepšie pochopiť zistený problém. Skoršie tímové porozumenie problému ho predurčuje k lepšej šanci na vyriešenie. Nie je možno praktické a ekonomicky prípustné vykonávať etapu vyhodnotenia počas každej iterácie v životnom cykle AWE procesu. Napokon aj tak etapou vyhodnotenia dosiahneme kratšie životné cykly a menej vynaložených vývojových prostriedkov na pomoc tímu porozumieť, čo koncový používateľ potrebuje.



Obrázok 1.5: Vytvorenie plánu vyhodnotenia v životnom cykle AWE procesu [MW01]

1.3.7 Iteratívny a inkrementálny vývoj

Iteratívne a inkrementálne vývojové cykly sú podstatné pre vykonávanie AWE procesu. Veľa web projektov začína etapou realizovateľnosti nasledovanou dlhou etapou analýzy a definovania požiadaviek. Pri predvídavom postoji, kde sú veľmi skoro vo vývojovom cykle zistené a zostavené (prevažne staticky) úlohy a navrhnuté riešenia, často pomáhajú charakteristiky projektu (rozpočet, časový rozmer, rozvrh, zdroje). Tieto charakteristiky uspokojia prvotné požiadavky vrcholového manažmentu.

Vďaka tomu sa často tím dostane do ilúzie, že vie všetko, čo potrebuje vedieť o žiadanom systéme. V takýchto projektoch obyčajne nasleduje návrh a implementácia navrhnutého systému s malou alebo bezvýznamnou biznis analýzou a analýzou požiadaviek žiadaného riešenia. Nedostatky v žiadanom riešení nie sú poriadne

identifikované pokiaľ sa neskončí testovanie. A bez poriadnej etapy vyhodnotenia problému nie sú spozorované, pokiaľ systém nezačne používať koncový používateľ.

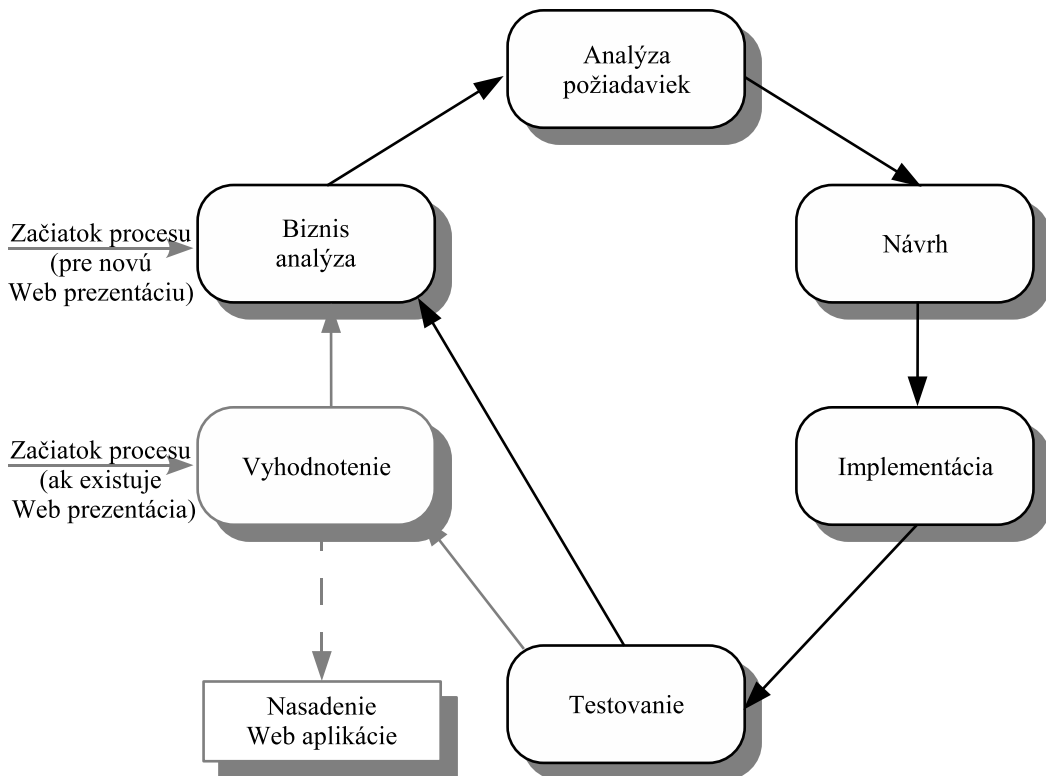
Mnohým softvérovým profesionálom je dobre známe, že rýchlejšie identifikovanie úloh počas životného cyklu systému je ekonomicky efektívnejšie. Čím dlhšie vývojárske tímy pokračujú s vývojom projektu, tým viac rozumejú problémovým miestam a efektívnejšie ich identifikujú v žiadanom systéme. Tento predvídavý postoj predstavuje vo vývoji web aplikácií najväčšiu výzvu na adaptovanie AWE procesu. Veľa organizácií a projektov potrebuje zamietnuť zotrvačnosť obklopujúcu web a softvérové vývojové aktivity, ak chcú zaviesť AWE proces do ich web inžinierskych projektov.

Predvídavý postoj pre rozsiahly vývoj web aplikácií je často zložitý, z dôvodu odlišnosti tímov, ktoré zahŕňajú analýzu, uskutočniteľnosť požiadaviek a ohraničenia, a ktoré vyvíjajú žiadané riešenie (návrh, implementácia, testovanie, vyhodnocovanie a nasadenie). Použitie rôznych ľudí pre projektovú analýzu a samotný vývoj zavádza komunikačnú bariéru pri vybudovaní žiadaného riešenia. Je preto rozhodujúce, aby boli všetci vývojári zahrnutí v každej etape AWE procesu.

Keď adaptujeme iteratívny a inkrementálny postoj pre vývoj softvérového systému, je podstatné určiť kritériá, podľa ktorých sa bude vyberať poradie úloh. Najviac stanovených výberových kritérií je zameraných na identifikáciu úloh, ktoré sú vnímané ako najrizikovejšie. Skutočne, AWE proces nemá výnimky. Každá iterácia by sa mala priblížiť k týmto najrizikovejším úlohám, ktoré by mohli spôsobiť neúspech projektu. Existujú riziká zabezpečujúce, že úsilie z predchádzajúcich iterácií nie je zmierené s inkrementálnym prírastkom vo vývojovej oblasti. Každá iterácia by sa mala zamerať na riešenie podmnožiny úloh predstavujúcich práve toto najväčšie riziko. Je tiež múdre zapamätať si, že zjednodušenie riešeného problému zjednodušuje budúcu zmenu a zlepšuje celkové navrhnuté riešenie.

Ako bolo spomenuté, je podstatné zahrnúť koncových používateľov v etape vyhodnotenia pred tým, ako môžeme presvedčivo povedať, že navrhnuté riešenie bude riešiť identifikovaný problém z biznis analýzy. Etapa vyhodnotenia s koncovým používateľom je však drahá a zaberá veľa času. To zavádza vážny konflikt s manifestom agilných riešení krátkych vývojových cyklov. Napokon neodporúčame etapu vyhodnotenia s konečným používateľom počas každej iterácie AWE procesu. Obrázok 1.6 opisuje typickú iteráciu AWE cyklu.

Vyhodnotenie by malo byť použité iba ak existuje čiastkový výsledok, na ktorom môžeme overiť identifikovaný problém a predpokladáme návratnosť investícií. Nedá sa predvídať, ako často je nutná etapa vyhodnotenia s koncovým používateľom. Výnimku tvorí posledná iterácia pred nasadením, kde je podstatné vyhodnotenie výsledku, aby sme mali istotu v nasadené riešenie. Obrázok 1.7 ukazuje iteráciu AWE cyklu obsahujúcu etapu Vyhodnotenie. Ostáva na zodpovednosti vývojára a organizácie posúdiť, ako často zahrnúť etapu Vyhodnotenie s účasťou koncového

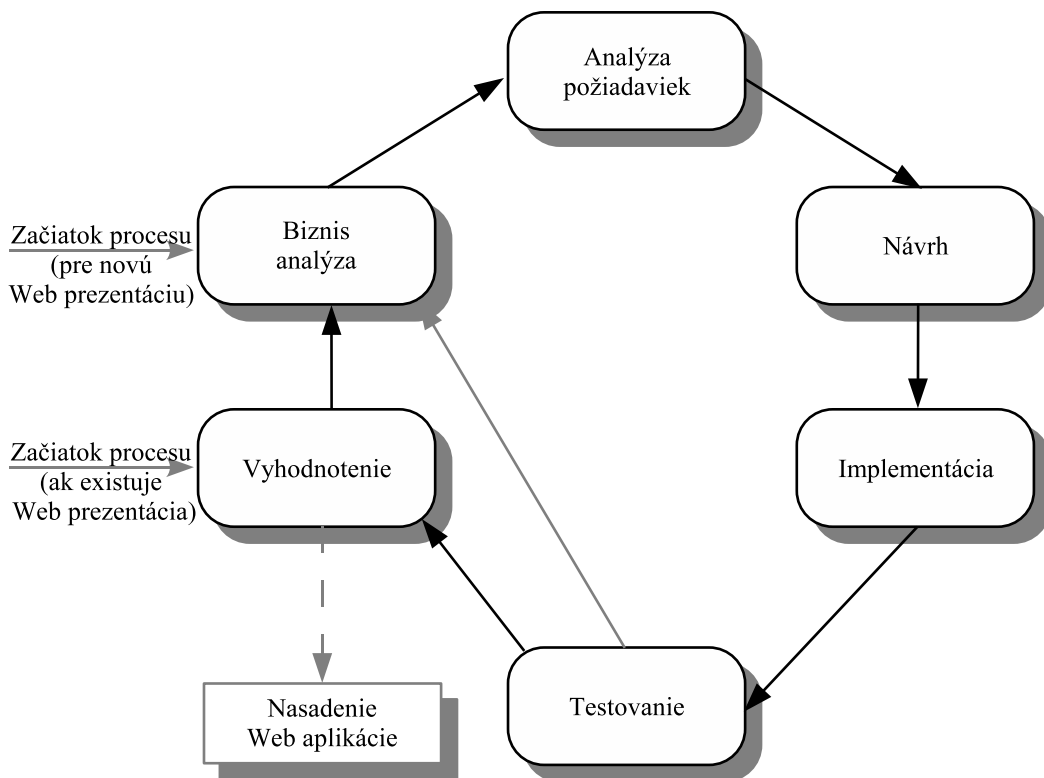


Obrázok 1.6: Opis typickej iterácie v životnom cykle AWE procesu.[MW01]

používateľa.

1.3.8 Začiatok AWE procesu

Existujú dve etapy, z ktorých môže začať cyklus AWE procesu: Biznis analýza a Vyhodnotenie. Ak sa začína vývoj novej web aplikácie, tak by sa mal cyklus AWE procesu začať etapou Biznis analýza. Ak ale existuje podobné riešenie, na lepšie porozumenie môže byť výhodné začať vyhodnotením konkurenčných aplikácií alebo iných web riešení. Ak sa rozvíja existujúca web aplikácia alebo riešenie, potom je múdre začať cyklus AWE procesu etapou Vyhodnotenie. Tam, kde existuje web aplikácia alebo web riešenie, je cenné použitie predchádzajúcich projektových výsledkov. Vstup do AWE procesu cez etapu Vyhodnotenie môže veľmi pomôcť v etape Biznis analýza. Obrázok 1.8 ukazuje možné vstupy do životného cyklu AWE

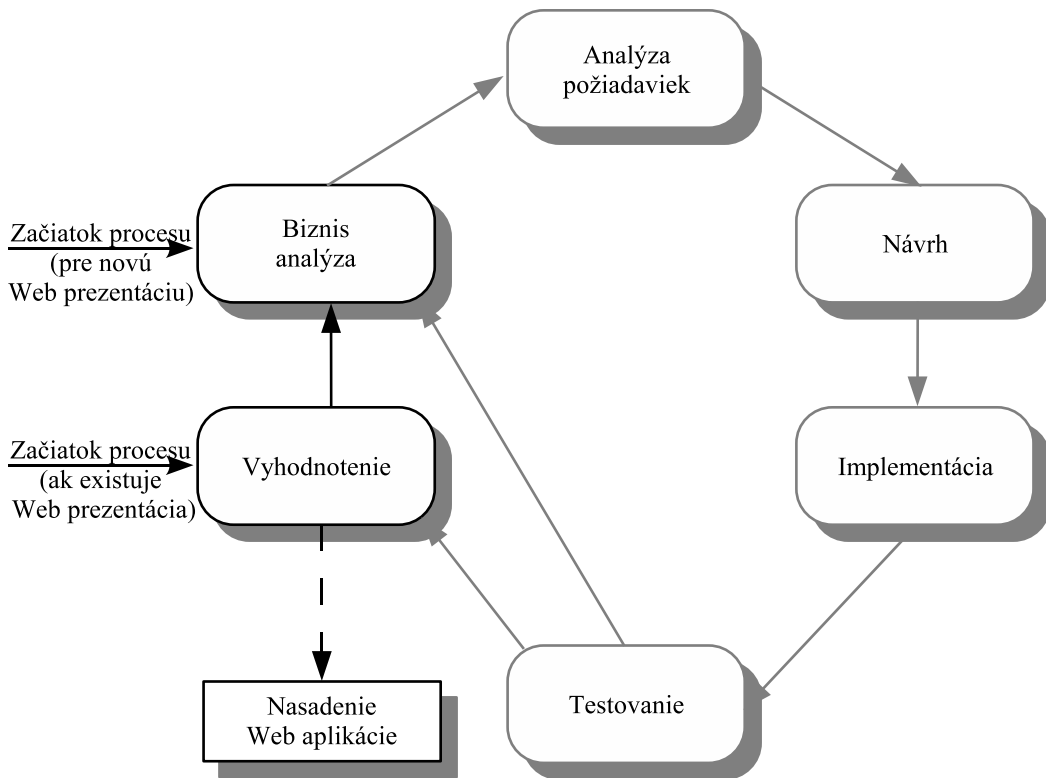


Obrázok 1.7: Opis iterácie v životnom cykle AWE procesu s etapou Vyhodnotenie. [MW01]

procesu.

1.4 Adaptívne, kreatívne vyhodnocovanie a prispôsobovanie techniky šité na mieru

Iteratívny a inkrementálny vývojový životný cyklus sa dobre hodí na vývoj web aplikácií. Na začiatku by mal mať každý projekt vyhodnotenú kritériá, ktorými sa budú identifikovať riziká, ktoré môžu byť odstránené nejakými metódami. Zo začiatku, počas prvých iterácií, sa musia vývojári zameriavať na najrizikovejšie prvky projektu. Každý tím a vývojár je zodpovedný za prispievanie a spoluprácu na úlohách spojených s úspešným integrovaním metód vhodných pre AWE proces. Prudké zmeny v biznis, doménovom alebo softvérovom modeli a veľký dopad



Obrázok 1.8: Etapy vstupov do AWE procesu. [MW01]

zmien na ostatné modely si vyžaduje trvalé sledovanie a vyhodnocovanie používaných metód. Je nevyhnutné, aby každý tím pravidelne sledoval prínos každej techniky (metódy) do projektu a to najlepšie po každej iterácii životného cyklu AWE procesu. Obrázok 1.9 znázorňuje typický iteratívny a inkrementálny vývojový životný cyklus AWE procesu s vyznačenými bodmi, kde by sa mali použiť vyhodnocovacie metódy. Iteratívny a inkrementálny štýl životného cyklu AWE procesu je založený na Boehmovom špirálovom modeli. Keď sa v etape vyhodnotenia zdá, že použitá technika pri vnútrotímovej komunikácii funguje, pokračuje sa v jej používaní, ale s uistením, že sú informované ostatné tímy prostredníctvom tzv. medzitímovej komunikácie. Tým sa zmenší špirálová krivka. Keď sa metóda zdá byť problematická, rozoberie sa, čo na nej funguje a čo na nej nefunguje. Keď členovia tímu veria, že pozmenením problematickej techniky dôjde k jej úspešnej adaptácii v rámci projektu, tak sa zmena aplikuje a technika sa naďalej používa.

V opačnom prípade sa jej treba zbaviť a nájsť inú, ktorá by ju v prípade potreby nahradila. Treba mať ale na pamäti, že je nutné informovať ostatné tímy pomocou medzitímovej komunikácie o uskutočnených rozhodnutiach o zmene metód a uviesť dôvody na objasnenie.

Príklad: Jeden tím sa rozhodol, že bude používať UML na modelovanie jednej AWE aplikácie. Zdalo sa, že UML bude ideálne, pretože má veľmi blízko k integrácii pomocou množstva CASE prostriedkov. Po prvej iterácii sa tím jasne zhodol, že UML nie je celkom vhodné pre modelovanie web aplikácií. Po následnom skúmaní objavili, že pre UML existuje rozšírenie pre web aplikácie s názvom „WAE for UML“. Členovia tímu sa zhodli na tom, že použijú WAE. Pri vyhodnocovaní po druhej iterácii životného cyklu AWE sa objavila oveľa väčšia spokojnosť s rozšírením „WAE for UML“ a tím sa ho rozhodol prijať ako projektový štandard. Všetky získané vedomosti o UML resp. „WAE for UML“ potom poskytli ostatným tímom.

1.5 WebE proces - Proces web inžinierstva

Na základe životného cyklu WebE procesu [Pre00] sú založené ďalšie dva varianty špirálového modelu. WebE proces obsahuje šesť fáz umožňujúcich ich paralelné vyvíjanie: Formulácia, Plánovanie, Analýza, Návrh, Generovanie stránok a Testovanie a Zákaznícke vyhodnotenie (obrázok 1.10).

Existujú dve časové verzie s rôznymi dĺžkami trvania, ktoré sa líšia činnosťami vykonávanými počas etáp Analýza, Návrh a Generovanie stránok a testovanie. Verzie sú pomenované nasledovne:

1. Dlhší procesný cyklus
2. Kratší procesný cyklus

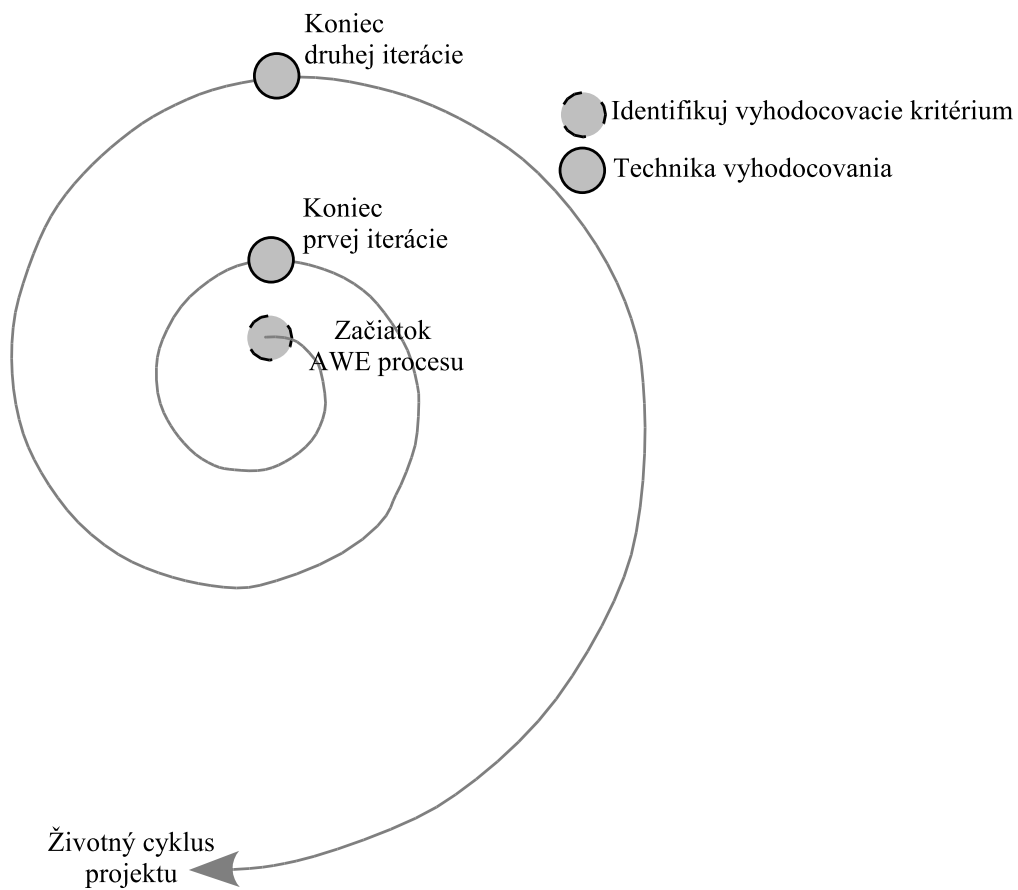
V nasledujúcich kapitolách charakterizujeme jednotlivé činnosti vykonávané v spomínaných etapách pre dlhší a kratší procesný cyklus.

1.5.1 Dlhší procesný cyklus

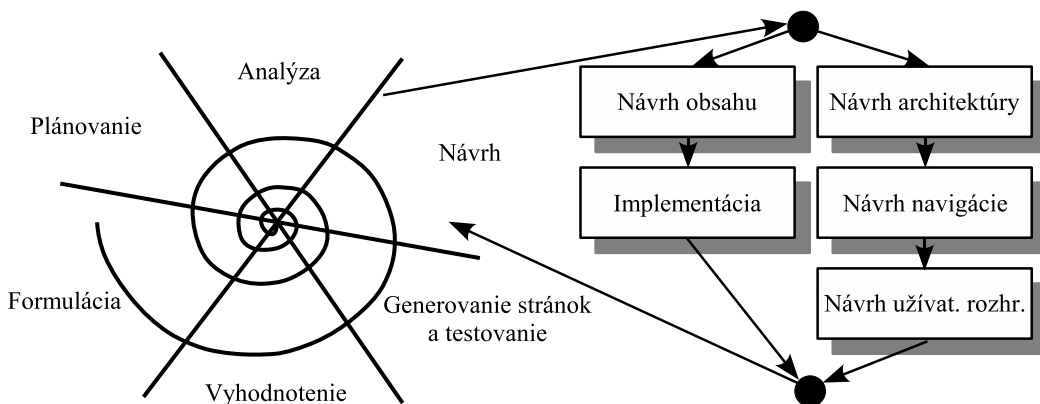
Činnosti v etape analýzy

Získanie požiadaviek - založených na potrebách organizácie, používateľov a zákazníkov. Získavanie požiadaviek môže byť vykonané rôznymi spôsobmi, ako sú napr. dotazníky, interview, pozorovanie, zbierka firemnej dokumentácie atď.

Analýza požiadaviek - kategorizácia a organizácia požiadaviek do súvisiacich skupín, preskúmanie vzťahov medzi požiadavkami, preskúmanie konzistencie,



Obrázok 1.9: Typický iteratívny a inkrementálny životný cyklus AWE procesu. [MW01]



Obrázok 1.10: WebE proces. [prob]

odstránenie nejednoznačností a pridanie pozabudnutí, ohodnotenie požiadaviek podľa potrieb zákazníka.

Analýza rizík - identifikácia, porozumenie a manažment rizík, pre čo najlepšie odhadnutie ich dopadu a v prípade vzniku problému zavedenie rezervného plánu.

Definícia požiadaviek - zhrnutie informácií získaných predošlými aktivitami do dokumentu, ktorý jasne zadefinuje množinu požiadaviek.

Špecifikácia požiadaviek - detailné opísanie systému požiadaviek, ktoré slúžia ako podklad pre vývoj systému. Môže sa tiež vyvinúť štandardná šablóna pre predvedenie konzistentnosti požiadaviek.

Systémové modelovanie - vývoj modelu systému, ktorý reprezentuje vstup, proces a kontrolovanie funkcií, výstup, návrh používateľského prostredia a akceptačných testov.

Validácia požiadaviek - overenie špecifikácie požiadaviek, ktoré zaistí, že všetky funkcionálne a nefunkcionálne požiadavky vyhovujú stanovenému procesu, projektu a produktu.

Manažment požiadaviek - identifikácia, kontrola a sledovanie zmien požiadaviek v prípade zmeny projektu.

Činnosti v etape návrhu

Architektonický návrh - mapovanie požiadaviek v architektúre web systému reprezentujúce štruktúru dát a programových komponentov.

Návrh navigácie - návrh navigačných ciest umožňujúci používateľom prístup k obsahu a službám web aplikácie podľa skupiny používateľov.

Návrh rozhrania - vývoj používateľských obrazoviek poskytujúcich efektívne komunikačné rozhranie medzi systémom a používateľom.

Návrh obsahu - návrh obsahu web aplikácie, ktorá obsahuje text, grafiku, efekty, obrázky, video a audio údaje.

Implementácia - vývoj web aplikácie založenej na informáciách a návrhoch získaných počas predchádzajúcich aktivít.

Činnosti v etape generovania stránok a testovania

Generovanie stránok - inštalácia web aplikácie a integrovanie do súčasného systému (ak je dostupný), ako aj adaptovanie na existujúce pracovné postupy.

Testovanie - testovanie jednotlivých modulov, ako aj celého systému metódou testovania čiernej skrinky a technikou testovania kódu.

Školenie - predvedenie systému a zaškolenie manažmentu a používateľov, vytvorenie on-line používateľskej príručky, referenčnej príručky a manuálu pre údržbu.

1.5.2 Kratší procesný cyklus

Činnosti v etape analýzy

Získanie požiadaviek - pomocou krátkotrvajúcich metód, ako sú interview a zbierka dokumentácie.

Analýza požiadaviek - kategorizácia požiadaviek a určenie ich dôležitosti podľa potrieb zákazníkov a používateľov.

Definícia požiadaviek - zhrnutie informácií získaných predošlými aktivitami do dokumentu a definovanie množiny požiadaviek poskytujúcich zmluvný dokument medzi manažmentom a vývojárom.

Činnosti v etape návrhu

Architektonický a návrh navigácie - abstraktné mapovanie požiadaviek a navigačných ciest do architektúry web systému.

Návrh rozhrania, obsahu a implementácie - rýchle prototypovanie rozhrania a jeho postupný vývoj do konečnej aplikácie spolu s implementovaním a začlenením.

Činnosti v etape generovania stránok a testovania

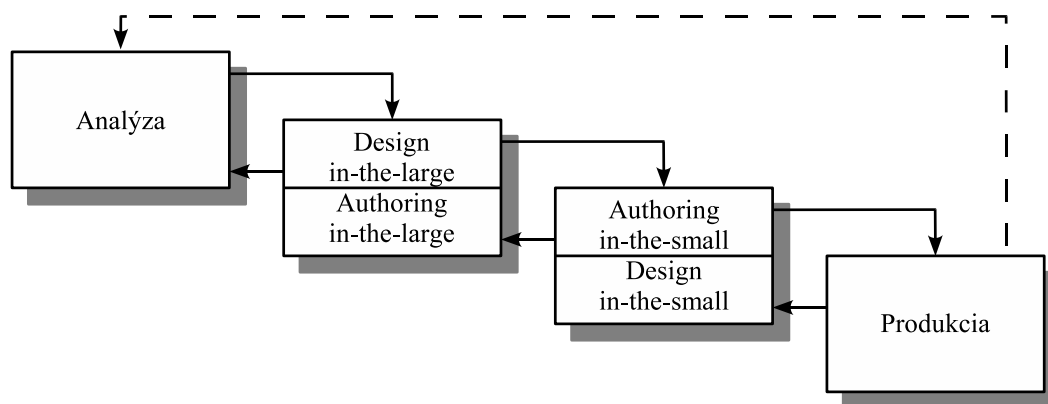
Generovanie stránok a testovania - inštalácia web aplikácie a jej otestovanie pomocou metódy čierna skrinka.

Školenie - krátke predvedenie systému, školenie manažmentu a koncových používateľov.

1.6 Hypertextový procesný model (Hypertext process model)

Ďalším modelom procesu vo web inžinierstve je Hypertextový procesný model (obrázok 1.11). Je zameraný na tvorbu komponentov za účelom modelovania funkcionálnych závislosti hypertextovej tvorby. Hlavné kroky Tvorba obsahu a Návrh hypertextu sú neoddeliteľne zviazané a sú tak zobrazené v nešpecifikovanom poradí alebo striedavo v niekoľkých krokoch. Obvykle tvorbe obsahu a návrhu veľkého rozsahu predchádza tvorba obsahu a návrhu malého rozsahu, ale závisí to z väčšej časti od osobného štýlu.

Jednotlivé kroky obsahujú množiny úloh a výsledkov. Počas analýzy, hlavne veľkých projektov, môžu byť napísané smernice za účelom definovať kvalitu, štandardy písania, cieľové skupiny, cieľové jazyky a dynamické aspekty zamýšľaného hypertextu. Tieto úvodné smernice môžeme porovnať s analýzou požiadaviek v softvérovom inžinierstve. Rozdelenie hypertextu do rozumných modulov a koordinácia spolupracujúcich pracovných skupín rozdelených vertikálne (modulovo orientované), alebo horizontálne (procesne orientované) sú potrebné na spracovanie veľkých projektov. Viacnásobné varianty hypertextu musia byť často plánované podľa rozdielných cieľových skupín (napr. rôzne jazyky alebo viaceré stupne bezpečného prístupu). Časovo obmedzená platnosť informácií môže požadovať zváženie viacerých verzií textu. Existujúci hypertext musí byť často integrovaný, prípadne prerobený do novej schémy a spôsobu navigácie. Pôvodnú schému poskytnutú grafickým dizajnérom je treba tiež zväžiť.



Obrázok 1.11: Hypertextový procesný model. [CL00]

Na základe analýzy sa v etape návrhu hypertextových vzorov „design-in-the-large“, vyrobia typy architektúr (uzly a odkazy), ktoré sa počas nasledujúcej etapy „design-in-the-small“ použijú ako šablóny. Zároveň vzor obsahu je definovaný pomocou „authoring-in-the-large“. Uzly a odkazy sú potom vytvorené reprezentujúcim obsahom dokumentu. Klasická úloha tvorby obsahu - formulovanie textu patrí do „authoring-in-the-small“. Už od návrhu navigácie a schémy sú spôsoby formulovania textu mapované a ich finálna formulácia je zjemňovaná počas všetkých týchto krokov.

Aby bolo možné schváliť vlastnosti výsledného hypertextu dosiahnutého automatickou výrobou, všetky kroky výrobného procesu by mali byť často testované, a to v každej etape. Len čo je špecifikovanie hypertextových dokumentov kompletne, výber príslušnej verzie, variantu a formátu by mal postačiť na generovanie finálneho hypertextu. Žiadna ďalšia zložitá tvorivá práca nie je potrebná.

1.7 Zhrnutie

Vývoj projektov v prostredí Internetu sa stal flexibilnejší a pozornejší na požiadavky zákazníka ako v klasických prostrediach softvérového inžinierstva. Zároveň pribudol dôraz na vzhľad používateľského rozhrania. Vzhľadom na spomenuté zmeny postupnou úpravou pôvodných procesov softvérového inžinierstva vznikli nové prispôbené procesy web inžinierstva. Do popredia sa dostal hlavne špirálový model. Inkrementálny a iteratívny vývojový životný cyklus, ktorý je založený práve na špirálovom modeli, sa dobre hodí na vývoj web aplikácií. Podľa veľkos-

ti, dôležitosti, formálnosti a požiadavky na rýchlosť dodania žiadaného systému, existuje na výber viac možných procesov.

Medzi formálnejšie procesy, ktoré udržujú štruktúru vývoja, patrí WebE proces, ktorý má dve časové varianty, dlhší a kratší. Výber jedného z nich závisí od veľkosti projektu a požiadaviek naň. Okrem formálnejších procesov sa do popredia dostáva agilný prístup k vývoju, ktorý určuje len určité oblasti, na ktoré by si vývojári mali dávať pozor, ale výber metód, ktorými to dosiahnu ponecháva na nich. Agilné metódy kladú prvoradý dôraz na splnenie požiadaviek zákazníka, spoluprácu s ním počas vývoja a umožňujú rýchle reagovanie na zmeny počas vývoja.

Jedinou oblasťou, kde neprenikol špirálový model, je spracovanie, návrh a tvorba hypertextu. Pre túto oblasť je navrhnutý Hypertextový procesný model, ktorý je založený na vodopádovom modeli.

Kapitola 2

Kategórie web aplikácií

V súčasnosti sa na Internete nachádza veľké množstvo rôznych web aplikácií. V tejto kapitole sa zaoberáme rozdelením týchto aplikácií do niekoľkých kategórií podľa rôznych kritérií. Hranice medzi kategóriami nie sú striktne určené a častokrát má konkrétna web aplikácia vlastnosti rôznych kategórií.

V nasledujúcej kategorizácii rozdeľujeme web aplikácie z používateľského hľadiska. Ku každej kategórii sú uvedené príklady existujúcich aplikácií i technológie, ktoré sa často používajú pri vývoji aplikácií konkrétnej kategórie. Kapitola vychádza z členenia podľa [GM01].

Z používateľského hľadiska môžeme web aplikácie rozdeliť nasledovne:

- informačné web aplikácie
- interaktívne web aplikácie
- vyhľadávacie web aplikácie

Názvy skupín sú charakterizované najčastejšie sa vyskytujúcimi web aplikáciami v danej skupine. Predchádzajúce členenie je príliš hrubé a preto sme jednotlivé skupiny ďalej rozdelili na kategórie. Zaradenie konkrétnych kategórií do skupín nie je jednoznačné. Jednotlivé kategórie sú charakterizované nezávisle od skupín a čitateľ si môže sám zaradiť kategórie web aplikácií do jednej zo spomenutých skupín.

Ako príklad uvádzame nasledovné rozdelenie:

- informačné web aplikácie
 - Informačné web aplikácie
 - Download web aplikácie

- interaktívne web aplikácie
 - Interaktívne web aplikácie
 - Transakčne orientované web aplikácie
 - Web aplikácie poskytujúce služby
 - On-line hry
- vyhľadávacie web aplikácie
 - Vyhľadávacie web aplikácie
 - Web aplikácie s prístupom k databáze

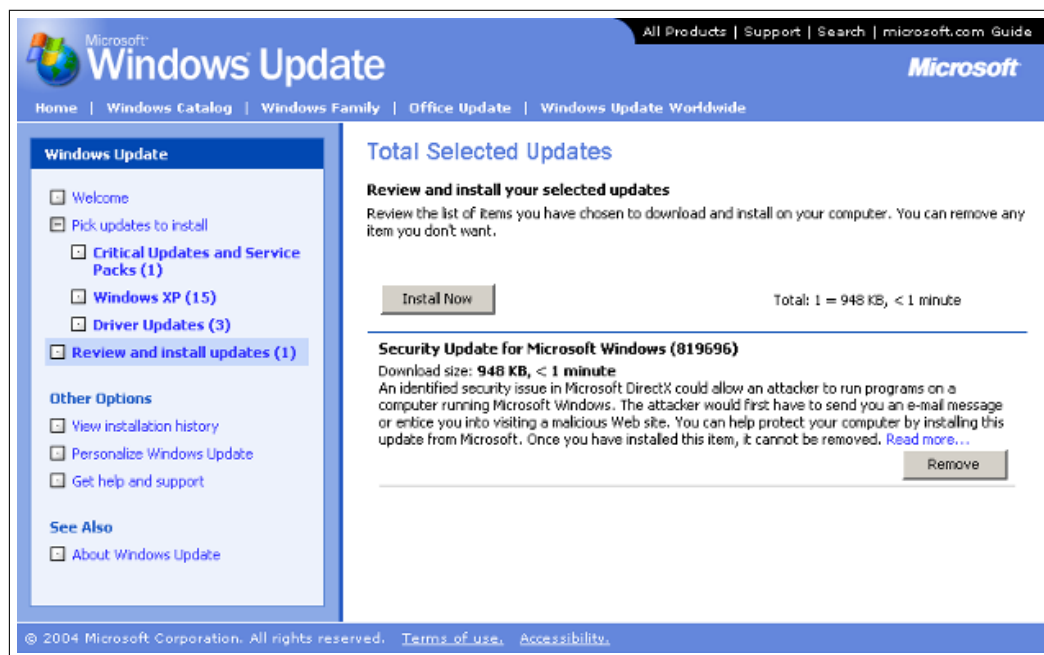
2.1 Informačné web aplikácie

Do kategórie informačných web aplikácií patria stránky on-line novín (www.sme.sk), spravodajských agentúr (www.cnn.com), katalógy produktov (katalog.avon.sk), servisné manuály, on-line knihy, firemné prezentácie. Väčšina obsahu je prístupná iba na čítanie. Používateľ prostredníctvom týchto typov web aplikácií získava nové informácie, novinky zo sveta, informácie o produktoch alebo informácie reklamného charakteru. V súčasnom Internete najviac aplikácií spadá práve do tejto kategórie. Na tvorbu týchto stránok, resp. web aplikácií sa využíva jazyk HTML na strane klienta, ak ide o menšie a jednoduchšie projekty. Jazyk HTML sa používa v kombinácii s jazykom na strane servera, ak je potrebné dynamicky generovať stránky s rovnakým vzhľadom, ale s rôznym obsahom. Využívajú sa tiež šablónové systémy, ktoré preddefinovanú šablónu naplňajú (často z databáz) rôznymi multimedialnými dátami (napr. text, obrázky, reklamy). Šablónové systémy najčastejšie nájdeme napr. pri on-line novinách alebo katalógoch produktov.

2.2 Download web aplikácie

Do kategórie download web aplikácií patria systémy, ktoré umožňujú používateľovi voľne alebo za poplatok skopírovať súbory rôzneho typu. Jedná sa napr. o tzv. FTP servery. Používateľ prostredníctvom takýchto serverov získava inštalačné súbory, hudbu, video alebo obrázky bez toho, aby musel opustiť svoj byt a zakúpiť si uvedené produkty v obchodoch. Taktiež do tejto kategórie patria servery s aktualizacími súbormi pre rôzne aplikácie, či už platené alebo neplatené. Typickým príkladom takéhoto systému je aplikácia firmy Microsoft - Windows Update (<http://v4.windowsupdate.microsoft.com/en/default.asp>), kde systém po jednoduchej komunikácii aj s neskúseným používateľom dokáže priamo

zistiť chýbajúce opravné súbory danej verzie operačného systému Windows a po odsúhlasení ich aj nainštalovať. Na vytvorenie takýchto systémov sa využíva protokol FTP alebo špecifická firemná technológia.



Obrázok 2.1: Web aplikácia firmy Microsoft pre aktualizáciu operačného systému Windows. <http://windowsupdate.microsoft.com>

2.3 Interaktívne web aplikácie

Do tejto skupiny web aplikácií patria systémy pre on-line aj off-line rozhovor, kladenie otázok širokému okruhu potenciálnych odborníkov v danej oblasti, rôzne súkromné kanály určené na komunikáciu, verejné fóra a podobné typy systémov. Aplikácie v tejto kategórii sa dajú ďalej rozčleniť na:

- programy na on-line komunikáciu (napr. ICQ, Windows Messenger)

Vo väčšine prípadov ide o jednoduchú komunikáciu či už textovou, zvukovou alebo aj obrazovou formou. Tento prostriedok komunikácie je zaužívaný najmä medzi mladými ľuďmi. Umožňuje rýchle a lacné riešenie na sprostredkovanie komunikácie na väčšie vzdialenosti s tým, že používatelia komunikujú

iba medzi sebou a teda ich rozhovor nie je verejne dostupný. Na implementáciu týchto systémov sa používajú rôzne programovacie jazyky ako napr. Java, alebo C++. Pre aplikáciu, ktorá sa vykonáva na strane klienta ako i rôzne databázy pre udržiavanie prihlasovacích mien a prístupových hesiel, ale aj pre ukladanie samotných príspevkov na strane servera.

- fóra (napr. `mojchat.atlas.sk`, `www.azet.sk`, `www.pokec.sk`)

Ide výhradne o komunikáciu v textovej forme. Väčšinou je k dispozícii zoznam miestností podľa rôznych diskusných tém. Používateľ sa môže rozhodnúť, či bude príspevok adresovaný priamo konkrétnemu používateľovi, alebo bude bez adresáta pre celú skupinu používateľov. Používateľ taktiež môže po odsúhlasení druhou stranou prejsť do tzv. tichého režimu, kedy je obsah rozhovoru prístupný len pre dvoch používateľov. Na realizáciu týchto systémov sa využívajú programovacie jazyky na strane servera (napr. ASP, PHP), ktoré riadia prístup k databáze príspevkov, používateľov a ďalších informácií s nimi súvisiacimi ako napr. rôzne štatistiky, koľko času používateľ strávil na diskusii, koľko príspevkov vytvoril. Dôležité je zabezpečiť dostatočne pravidelné a rýchle obnovovanie stránky, aby zostala zachovaná aktuálnosť príspevkov.

- odborné fóra

Ide taktiež o komunikáciu výhradne v textovej forme. Komunikácia má istý odborný charakter podľa zamerania stránky, na ktorej sa daný systém nachádza. Častokrát sú takéto fóra pripájané k odborným i neodborným článkom a používatelia diskutujú na tému opísanú v článku. Existujú i fóra, v ktorých môže používateľ opísať problém, ktorý rieši a iný používateľ, ktorý má v danej oblasti dostatočné skúsenosti mu poradiť. Časté sú fóra takéhoto typu na témy programovania v rôznych programovacích jazykoch. Celá komunikácia medzi používateľmi je verejná, prípadne obmedzená na prihlásenie sa do systému. Používateľ môže reagovať na konkrétny príspevok alebo pridať nezávislý príspevok. Vzhľadom na časovú nepravidelnosť reakcií na príspevok je vo väčšine takýchto fór možnosť nastaviť upozornenie na odpoveď prostredníctvom elektronickej pošty. Podobne ako u klasických fór i tu sa využívajú jazyky na strane servera a databázy pre udržiavanie informácií o príspevkoch, používateľoch a iných (rôzne štatistiky). Nekladie sa dôraz na rýchle pridávanie príspevkov. Vloženie príspevku môže trvať i niekoľko minút.

View Message View		Per page 25
New thread		
Subject	Author	Date
Great tool! Spy++ suggestion	Don Kackman	9:12 7 May '04
using for all windows	ploufs	21:47 20 Feb '04
Control	olinat	22:21 22 Sep '03
Re: Control	neeleshd	7:29 20 Feb '04
Integrate with ControlInspector?	swythan	6:00 21 Aug '03
Re: Integrate with ControlInspector?	Jabes	6:32 21 Aug '03
Re: Integrate with ControlInspector?	Rama Krishna	9:30 22 Aug '03
Cool	Stoyan Damov	3:48 21 Aug '03
I like it!	Holger Persch	1:46 21 Aug '03
Classic	Kant	19:10 20 Aug '03
Re: Classic	Rama Krishna	10:13 21 Aug '03
Re: Classic	Kant	21:59 21 Aug '03
Excellent!	jdunlap	15:48 20 Aug '03

Last Visit: 1:55 Friday 28th May, 2004

Obrázok 2.2: Odborné fórum na www.codeproject.com

2.4 Vyhľadávacie web aplikácie

V súčasnosti medzi najčastejšie navštevované web aplikácie patria práve vyhľadávacie systémy (napr. www.google.com, www.yahoo.com, www.zoznam.sk). Používateľ sa za pomoci týchto systémov môže jednoducho dostať k odkazom, na ktoré by prirodzeným spôsobom prišiel len ťažko. Používateľ špecifikuje svoje konkrétne požiadavky na vyhľadávaný reťazec a po potvrdení systém prechádza svojimi databázami odkazov a vyhľadáva hľadaný reťazec.

Pri väčšine takýchto systémov je možné vybrať si medzi dvomi režimami zadávania požiadaviek na hľadaný reťazec a to jednoduché vyhľadávanie a rozšírené vyhľadávanie. Pri jednoduchom vyhľadávaní používateľ môže zadať iba vyhľadávaný reťazec bez iných možností špecifikácie. Pri rozšírenom vyhľadávaní je možné zadať napr. jazyk, v ktorom má byť nájdená stránka, formát súboru, v ktorom má byť hľadaný reťazec, presnú frázu, ktorá má byť hľadaná, výskyt reťazca (v názve stránky, v tele stránky atď.), dátum poslednej modifikácie (reťazec je vyhľadávaný iba v stránkach, ktoré boli modifikované najneskôr v danom dátume), počet zobrazených odkazov a iné požiadavky zužujúce vyhľadávanie. Požiadavky pri rozšírenom vyhľadávaní môžu byť ľubovoľne kombinované. Pre vytváranie takýchto systémov sa používajú komplikované algoritmy pre vyhľadávanie dát v rozsiahlych databázach (napr. vyhľadávací systém www.google.com tvrdí, že v súčasnosti vyhľadávanie prechádza až 4 285 199 774 web stránok).



Obrázok 2.3: Vyhľadávacia web aplikácia na www.google.com

2.5 Transakčne orientované web aplikácie

Jedná sa väčšinou o objednávkové služby s platením platobnou kartou, v hotovosti pri doručení alebo iným spôsobom. Systémy identifikujú používateľa podľa predchádzajúcej on-line registrácie alebo podľa písomnej registrácie (napr. zákaznícka zóna veľkoobchodov väčšinou vyžaduje predchádzajúcu osobnú registráciu u predajcu). Tieto systémy využívajú rôzne prostriedky informačných technológií na dosiahnutie želaného efektu. Na najnižšej vrstve sú databázy výrobkov a ponúkaných služieb. Nad databázami je implementované na jednej strane prostredie pre predajcov s funkcionalitou pre definovanie a dopĺňanie nových výrobkov, služieb, vlastností výrobkov, atribútov výrobkov, sledovanie štatistík používateľov. Túto podmnožinu systémov ovládajú vyškolení ľudia. Ide o netriviálne mnohoparametrové systémy, ktoré môžu byť riešené formou klasickej aplikácie alebo cez web rozhranie.

Na druhej strane sú prostredia pre zákazníkov riešené formou web aplikácií. Aplikácie musia byť dostatočne jednoduché a prehľadné, zobrazujú a kategorizujú výrobky podľa rôznych preddefinovaných alebo používateľom definovaných kritérií. Sprevádzajú používateľa viacerými krokmi, ktoré sú potrebné

na realizáciu objednávky. Ide o vyhľadanie výrobku, prezeranie a porovnávanie vybraných parametrov výrobku s výrobkom inej značky, prispôsobenie výrobku (`www.compaq.com` - používateľ si môže poskladať vlastný notebook z predpripravených komponentov), výber výrobku, identifikácia používateľa (či už na základe registrácie alebo formou mena a adresy, kam bude objednávka doručená), výber spôsobu platby, odoslanie alebo zápis objednávky pre spracovanie. Poradie týchto krokov nie je striktné dané. Existujú aplikácie, pri ktorých je nutné prihlásiť sa hneď na začiatku objednávky, neprihlásený používateľ nemá možnosť ani len prezerat' a porovnávat' rôzne výrobky.

Web aplikácie tohto typu sú nad databázami rôznych komerčných výrobcov vytvárané pomocou jazykov na strane servera, šablónovými prostriedkami pre štandardizovanie používateľského prostredia a oddelenia dát od používateľskej prezentácie. Typickými príkladmi sú `www.amazon.com`, `www.compaq.com` (časť pre on-line objednávky).

2.6 Web aplikácie poskytujúce služby

Často ide o jednoduché výpočtové aplikácie postavené na základných matematických operáciách. Používateľ do vstupného formulára zadá niekoľko základných dát, ktoré sú použité pre ďalšie spracovanie. Kontrola i spracovanie vstupných parametrov môže prebiehať viacerými spôsobmi podľa toho, kde je samotná operácia realizovaná:

- operácia sa vykonáva na klientskej stanici
 - využíva sa Java, VBS, JavaScript, Flash a ďalšie
 - pri použití VBS alebo jazyka JavaScript je možné zistiť spôsob, akým sú dáta spracovávané
 - jednoduché matematické operácie, málokedy využívajú dáta z on-line databáz
- operácia sa vykonáva na serveri
 - spracovanie sa vykonáva v rámci skriptov na strane servera
 - sú známe iba výsledky operácií, nie ich priebeh
 - všetky medzivýsledky a chybové správy sa posielajú klientovi vo forme web stránok
- kombinovaný spôsob

- je často využívaný
- kontrola vstupných dát, pokiaľ je to možné, sa vykoná na klientskej strane (kontrola typov, kontrola správneho zadania adresy, dátumu atď.), chyba vo vstupných dátach sa častokrát oznámi používateľovi vo forme okna so správou
- výpočty a zložité operácie sa vykonávajú na strane servera, výsledky sú klientovi doručené vo forme novej web stránky.

The screenshot displays a real estate listing page with a mortgage calculator on the right. The listing details include:

- Reklama:** Advertisement for 'Isn't it about TIME?' with a 'CLICK HERE' button and 'Polyfonické zvonenia ZADARMO!!!'.
- Property Details:**
 - I 130,31 m², 2^o loggia po 6,24 m² cena: 5 110 717 Sk
 - G mezonet 125,09 + 6,24 loggia, 58,15 terasa cena: 5 529 668 Sk
 - H mezonet 103,88 + 6,24 loggia, 58,24 terasa cena: 4 723 062 Sk
 - 4 izbové
 - K 135,94 m² + balkón a terasa 64,58 m² cena: 5 945 097 Sk
 - 18 x garážové státie - cena : 290.000 Sk+DPH
- V CENE ZAHNUTÉ:** samostatné plynové kúrenie / alebo centrálna/, plastové okná, dlažby, obklady, plávajúce parkety - podľa vlastného výberu, dvere a drevené zárubne, kovanie na dvere, kompletná sanita - umývadlo, vaňa a/sprch.kút - podľa výberu, WC, vodovodné batérie, bezpečnostné dvere vonkajšie protipožiarne, v cene DPH 19%, podiel na pozemku a spoločných častiach a zariadeniach domu. Možnosť príkúpenia garáž.státia.
- Podnikateľsky priestor:**
- PRÍZEMIE S VÝKLADOM NA ULICU:**
- OP1 58,01m² cena: 25 000 Sk/m² + DPH
- OP3 65,65m² cena: 25 000 Sk/m² + DPH

The mortgage calculator on the right includes:

- Kalkulátor pôžičky:**
 - Úroková sadzba: 5,40 %
 - Dĺžka úveru (v rokoch): 20
 - Výška úveru: 32000 Sk
 - Mesačná splátka: 218 Sk
 - Prepočítaj
- Reaguj na inzerát:**
 - komu: valent@reality-servis.sk
 - meno: [input field]
 - kontakt: [input field]

Obrázok 2.4: Kalkulátor pôžičky na reality.sme.sk

2.7 Aplikácie s prístupom k databáze

Existuje veľké množstvo databáz a tiež web aplikácií, ktoré tieto databázy využívajú. Ak sa pozrieme na zložitejšie systémy, za každým z nich stojí väčšia alebo menšia databáza a oprávnené by tieto systémy mohli byť zaradené do tejto kategórie. Potom by sa ale členenie zjednodušilo na aplikácie s databázovou podporou a aplikácie bez nej. Pod aplikáciami v tejto kategórii budeme rozumieť prevažne

jednoduché web aplikácie využívajúce (nie zložité) databázy k priamemu zobrazeniu požadovaných dát. Typickým príkladom môže byť aplikácia obsahujúca základné informácie o mestách v danom štáte ako je názov, stručný opis, počet ľudí, umiestnenie, smerové poštové číslo, telefonickú predvoľbu. Používateľ zadá cez jednoduchý formulár kritériá zobrazenia a systém mu poskytne aktuálne údaje z databázy.

Tieto aplikácie sa dajú ďalej rozdeliť na platené a neplatené. Ďalej si uvedieme výhody a nevýhody z toho vyplývajúce:

- platené aplikácie
 - sú prístupné používateľovi až po častokrát zdĺhavej registrácii
 - podľa požadovanej sumy sú dáta aktuálne (so zvyšujúcimi poplatkami rastie aktuálnosť dát)
 - výškou poplatkov je garantovaná aj funkčnosť služby na požadovanú dobu a požadovaný čas
- neplatené aplikácie
 - prístupné bez akejkolvek registrácie alebo len po jednoduchej registrácii (často stačí uviesť elektronickú adresu)
 - často nie je zaručená aktuálnosť dát
 - veľké množstvo používateľov môže spôsobiť výpadky a znefunkčnenie systému.

Na realizáciu aplikácií tohto typu sa používajú jazyky na strane servera spolu s (často voľne dostupnými) databázami. Pre zobrazenie vstupných formulárov a získaných informácií sa na strane klienta najčastejšie používa jazyk HTML, prípadne JavaScript.

2.8 Dátové sklady

Mnohé z problémov pri prístupe k dátam riešia práve dátové sklady. V súčasnosti neexistuje veľa dokumentov, ktoré by spájali Internet, resp. web aplikácie s dátovými skladmi, hoci zvyšujúce sa nároky na vyhľadávanie a zväčšujúce sa množstvá prehľadávaných a sprístupňovaných dát sú dobré predpoklady pre spojenie týchto dvoch technológií.

Dátové sklady sprístupňujú dáta z rôznych databáz (aj fyzicky vzdialených) a poskytujú používateľovi vyhľadávané dáta. Z používateľského hľadiska sú tieto systémy takmer nerozoznatelné od systémov s prístupom do jedinej databázy

(možno až na o niečo dlhšiu dobu prístupu). Internet poskytuje pre aplikácie využívajúce dátové sklady možnosť zverejnenia veľkej skupiny dát poskladaním z rôznych databáz a následne uniformnú prácu s nimi.

2.9 Voľne dostupné služby

Služby aplikácií tohto typu často využívajú študenti z dôvodu ich ceny a ľudí, ktorí sa nedostávajú pravidelne k Internetu a pracujú na rôznych počítačoch (napr. internetové kaviarne, škola, internát). Do tejto kategórie zaraďujeme neplatené služby typu webmail, webhosting, štatistiky a iné. Služby tejto kategórie sú dostupné z ktoréhokoľvek počítača pripojeného na Internet, nie sú viazané na konkrétny počítač, resp. na konkrétne prostredie. Napr. elektronická pošta vo forme aplikácie webmail je prístupná odkiaľkoľvek na svete a nie len z prostredia firmy, kde sa nachádza poštový server resp. z konkrétneho počítača, ktorý je nakonfigurovaný na konkrétne konto elektronickej pošty.

2.9.1 Webmail aplikácie

Aplikácia umožňuje používateľovi po prihlásení sa do systému a predchádzajúcej registrácii cez ľubovoľný web prehliadač pristupovať k svojmu osobnému kontu, prijímať poštu, odosielať poštu, vytvárať priečinky a triediť prijatú poštu, aktivovať a deaktivovať ľubovoľné filtre (napr. automatické preposielanie správ na inú adresu, nastavenie automatickej odpovede, automatické triedenie prijatej pošty podľa rôznych kritérií), vytváranie zoznamu kontaktov a podskupín kontaktov.

V súčasnosti väčšina webmail aplikácií umožňuje i aktiváciu filtra nechcených správ a ďalšie pomôcky uľahčujúce prácu s veľkým množstvom prijatých správ. Väčšina poskytovateľov voľne dostupných webmail aplikácií umožňuje používateľom skopírovať si prijaté správy do klientských aplikácií (napr. MS Outlook) pomocou protokolu POP3, už menej z nich umožňuje odosielanie správ pomocou protokolu SMTP z daného konta.

2.9.2 Webhosting

V prípade, že chce používateľ publikovať na Internete akékoľvek informácie musí využiť webhosting. V súčasnosti sú dva spôsoby webhostingu.

- registrácia domény u autorizovanej spoločnosti
 - za služby spoločnosti treba zaplatiť (často iba jednorázový poplatok)

- v prípade využitia aj diskového priestoru spoločnosti pre fyzické umiestnenie stránok sú poplatky periodické a ich výška závisí od poskytovaných služieb a veľkosti poskytnutého priestoru
- majiteľ konta získa celú doménu (`www.nazov_stranky.sk`, resp. `www.nazov_stranky.skratka_krajiny`)
- využitie bezplatného webhostingu
 - využitie týchto služieb je vo väčšine prípadov zadarmo (napr. u `www.host.sk` sa za každé nové konto platí jednorázový poplatok 1000 Sk)
 - majiteľ konta získa iba subdoménu a nie doménu
 - adresa vytvorených stránok vyzerá najčastejšie `www.nazov_konta.nazov_poskytovateľa.skratka_krajiny` (`www.host.sk`), resp. `www.nazov_poskytovateľa.skratka_krajiny/nazov_konta` (`www.atlas.sk` resp. `www.mojweb.sk`)
 - častokrát je súčasťou poskytovaných služieb i vytvorenie cez databázy ovládanej web rozhranie. Túto databázu môžu využívať skripty daného používateľa (napr. `www.host.sk`)

U oboch typov webhostingu používateľ získa prístup k svojmu web priestoru najčastejšie prostredníctvom protokolu FTP alebo web rozhrania. Pre vytváranie spomenutých web aplikácií sa využívajú častokrát technológie ako napríklad ASP alebo JSP. Výnimkou nie je ani použitie voľne dostupných technológií ako PHP. Pre udržiavanie informácií získaných registráciou používateľov sa využívajú rôzne typy databáz.

Väčšina poskytovateľov voľne dostupných služieb poskytuje aj bezplatný web-mail aj bezplatný webhosting. Typickými príkladmi týchto web aplikácií sú `www.atlas.sk`, `www.host.sk`, `www.pobox.sk` a iné.

2.10 On-line hry

Aplikácie tohto typu sa dajú ďalej rozdeliť do dvoch kategórií. Prvou sú hry, ktoré k svojej činnosti potrebujú na Internete herný server, na ktorom sa stretávajú skupiny hráčov a súťažia medzi sebou. Medzi tieto hry patria napríklad Internet Hearts, Internet Reversi (štandardne dodávané s Windows XP Professional). V tomto prípade ide o klasické hry vyvíjané štandardnými spôsobmi, ktoré využívajú služby internetového servera.

Druhou skupinou sú aplikácie vytvorené prevažne v jazyku Java alebo Flash. Zahrňajú jednoduché programy zábavného charakteru. Kód samotnej hry sa vyko-

náva na pracovnej stanici používateľa a potrebné súbory hry sa pred jej spustením nakopírujú zo servera poskytujúceho danú hru.

Čo sa týka hier, ide prevažne o paródie klasických hier, zosmiešnenia rôznych životných situácií, hoci sa medzi nimi nájdu i originálne nápady. Aplikácie z tejto kategórie sú určené najmä, no nielen nižším vekovým skupinám. Typickým príkladom je hra, ktorá spôsobila nemalé ekonomické škody na celom svete (www.yetisports.net), pri ktorej pracovníci viacerých firiem zanedbávali svoje povinnosti.

Na Slovensku túto skupinu web aplikácií reprezentujú aplikácie prístupné na www.onlinehry.sk. Web aplikácie tohto typu sú väčšinou malého formátu a grafickým spracovaním sa nedajú porovnávať s klasickými hrami. Na druhej strane on-line hry často ponúkajú až nadštandardnú možnosť hry viacerých používateľov po sieti, kedy používatelia medzi sebou súperia cez Internet, oddelení i tisíckami kilometrov. On-line hry sú jednou zo skupín web aplikácií, ktorá si v poslednom období získava stále väčšiu a väčšiu popularitu. Dôkazom toho sú aj výsledky výskumu [Net04], podľa ktorých popularita on-line hier vzrástla v Európe medzi januárom 2002 a januárom 2003 až dvojnásobne. Medzi typické on-line hry neodmysliteľne patria kasíno, ruleta, šach alebo biliard.



Obrázok 2.5: On-line hra na www.yetisports.net

Kapitola 3

Návrh web aplikácií

V tejto kapitole priblížime návrh web aplikácií z rôznych pohľadov. Objasníme myšlienku použitia hypermedií a s nimi spojenými návrhovými metódami. Načrtujeme použitie návrhových vzorov pri tvorbe web aplikácií a myšlienky znovupoužitia.

3.1 Hypermediá

V súčasnosti je Internet obrovským zdrojom informácií nielen textového charakteru, ale poskytuje aj ďalšie typy médií ako obraz, zvuk, video, animácie alebo aj programy. Vďaka týmto rôznym typom možno web aplikácie z pohľadu prezentovania informácií charakterizovať ako hypermediálne, teda kombinujúce hypertext a multimédiá. Použitie odkazov v hypertexte poskytuje prirodzený a efektívny prístup k informáciám vďaka asociatívnemu charakteru ľudskej mysle. Kombináciou viacerých mediálnych typov možno u používateľa vyvolať lepší zážitok spojený s efektom ľahšieho zapamätania si informácií.

Hypermediá možno na jednej strane považovať za mechanizmus pre manažovanie rozsiahlych a komplexných informácií. Na druhej strane možno hypermediá považovať aj za nástroj pre poskytovanie prístupu k týmto informáciám, umožňujúc tak rôzne pohľady a zobrazenia pre rôznych používateľov. Na tento účel sa kombinujú metódy navigácie, používateľských rozhraní, odkazov a dátových uzlov a.i.

Pri hypermediálnych systémoch možno identifikovať dva základné modely: *reprezentačný model* a *návrhový model* [RSG97]. Reprezenačný model poskytuje mechanizmus na modelovanie štruktúry dokumentov pre ich vzájomnú výmenu medzi autorskými nástrojmi. Boli vytvorené dva základné reprezentačné modely hypermedií:

Dexterov model hypermédií [HS94] - Definuje dôležité abstraktné črty identifikované v množstve existujúcich hypermediálnych systémoch. Snahou je možnosť porovnania funkcionality rôznych systémov a zvýšenie interoperability. Definuje niekoľko úrovní, ktoré formalizuje pomocou špecifikačného jazyka Z.

Amsterdamský model hypermédií [HBv94]- Rozširuje Dexterov model hypermédií o časové aspekty, použitie dynamických médií ako zvuk a video.

Reprezentačné modely hypermédií sa používajú najmä na tvorbu prenositeľných autorských prostredí. Návrhové modely sú aplikačne orientované a poskytujú formálny prístup k návrhu hypermediálnych systémov.

Vývoj hypermediálnych systémov zahŕňa členov tímu s rôznymi profesiami, nielen programátorov, ale aj autorov, dizajnérov a grafikov, umelcov, hudobníkov. Vývoj tradičného softvéru zvyčajne potrebuje len IT špecialistov so znalosťami špecifickej oblasti a špecifickými nástrojmi. Tvorba hypermediálnych produktov si však vyžaduje použitie rôznych nástrojov na manipuláciu s multimediami objektami a navigačnými prostriedkami ako autorské nástroje, zvukové a video editačné programy a pod.

3.2 Modely a metódy návrhu web aplikácií

S postupným rozširovaním Internetu vzniká množstvo web aplikácií a systémov. Často však boli a sú vytvárané ad-hoc, bez štruktúry a efektívneho návrhu. Výsledné aplikácie boli slabej kvality s ťažkou údržbou, minimálnou možnosťou pre znovupoužitie a neodzrkadľovali potreby používateľov. Použitie tradičných metód softvérového inžinierstva sa ukázalo ako neadekvátne vzhľadom na estetické a kognitívne aspekty dôležité pri návrhu hypermediálnych systémov. V dôsledku týchto problémov sa začali vytvárať metódy, ktoré poskytovali systematický a disciplinovaný prístup k návrhu, dokumentovaniu a údržbe rozsiahlych web aplikácií. Nové metódy prihliadajú na špecifické aspekty hypermédií ako sú odkazy, navigácia, rôzne typy médií a pod.

Štruktúrované návrhové metódy a modely napomáhajú autorom počas vývoja web aplikácií. Použitie návrhových metód a modelov môže priniesť výhody ako:

- oddelenie nezávislých častí návrhu
- dobre definované kroky napomáhajúce vývojárom v postupe na projekte
- podpora objektovo orientovaných, relačných, ale aj iných prístupov
- návrhové diagramy pre znovupoužitie

- návrhové metódy podporiteľné CASE nástrojmi
- otvorenosť pre implementáciu

Nové metódy a modely majú však aj niekoľko nedostatkov:

- metódy predpokladajú isté znalosti softvérového inžinierstva, ako objektovo orientovaný návrh, návrh databáz a pod.
- obmedzujú sa na aplikácie s centralizovanými databázami
- vyžadujú viac času a úsilia pri návrhu
- existuje málo podporných CASE nástrojov založených na týchto metódach

Niektoré nové návrhové metódy vychádzajú z princípov návrhu databáz, alebo objektovo orientovaného návrhu. Možno však pre ne identifikovať spoločné fázy návrhu:

- konceptuálny návrh
- návrh navigácie
- návrh používateľského rozhrania

Tieto fázy súvisia so snahou o modelovanie hypermediálnych aplikácií na troch nezávislých úrovniach: *fyzickej, logickej a prezentačnej*. Cieľom je nezávislosť medzi dátami a aplikáciou a nezávislosť medzi aplikáciou a prezentáciou. Nezávislosť medzi dátami a ich prezentáciou vyžaduje logický opis nezávislý od spôsobu ako sú údaje poskytované používateľovi. To umožňuje zmenu formátu prezentácie dát a zmenu navigačnej štruktúry bez zmeny spôsobu uloženia dát. Nezávislosť medzi dátami a aplikáciou umožňuje použitie dát v iných aplikáciách. Nezávislosť medzi aplikáciou a prezentáciou umožňuje tvorbu platformovo nezávislých a portabilných aplikácií.

V nasledujúcich kapitolách sa budeme venovať najznámejším novým návrhovým metódam hypermediálnych aplikácií. Priblížime Hypertext Design Model [GPS93], Object Oriented Hypermedia Design Methodology [SRB96] a The Relationship Management Methodology [ISB95]. Okrem týchto však boli vyvinuté aj ďalšie metódy, z ktorých môžeme spomenúť Enhanced Object-Relationship Model [Lan94], WebArchitect [Tak97] alebo HM Data Model [MK93].

3.2.1 Hypertext Design Model (HDM)

HDM [GPS93] model je vhodný pre „návrh vo veľkom“. Model sa zameriava na reprezentáciu informačných objektov a navigačných štruktúr komplexných aplikácií bez konkretizovania implementačných detailov, čo ho robí systémovo a nástrojovo nezávislým. HDM poskytuje návod, ktorý vedie návrhára a napomáha mu zachytiť dôležité aspekty a štruktúry navrhovanej aplikácie.

HDM je založený na princípoch návrhu databáz. Vychádza z entitno-relačného modelu, ktorý rozširuje o hierarchickú organizáciu. Na návrh štruktúry aplikácie sa používajú primitívy. Za základné primitívy sa považujú *entity*, ktoré reprezentujú reálne objekty sveta. Nad týmito *entitami* sa ich abstrakciou vytvára *typ entity*, ktorý charakterizuje spoločné vlastnosti rôznych *entít*. Tento prístup možno nájsť aj pri tvorbe entitno-relačného modelu alebo pri objektovo orientovanom návrhu. Samotná *entita* sa skladá z viacerých *komponentov*, pričom *komponent* sa opäť môže skladať z menších *komponentov*. Samotný *komponent* sa skladá z jednej alebo viacerých *jednotiek*, ktoré sú atomické. *Jednotka* reprezentuje samotnú informáciu, pričom môže byť rôzneho mediálneho typu.

Významnou črtou HDM je zavedenie *perspektív*. Hlavnou myšlienkou je prezentovať tú istú informáciu z rôznych pohľadov podľa potrieb používateľa. Ako príklad možno uviesť opis nejakého stroja textom, obrazom alebo videom, alebo tiež opis v rôznych jazykoch. Perspektívy možno zaviesť do ľubovoľnej úrovne vyššie uvedenej hierarchie.

Na opis vzájomných vzťahov v hierarchickej organizácii primitív a v ich vzťahu k rôznym perspektívam boli zavedené rôzne druhy odkazov:

- odkazy medzi perspektívami - pre spojenie viacerých perspektív tej istej informácie
- odkazy medzi primitívami - pre spojenie primitív do hierarchickej štruktúry
- odkazy medzi aplikáciami - návrhárom voľne použiteľné odkazy podľa potreby aplikácie alebo doménovej oblasti, pre rôzne druhy prepojení

HDM nezavádza žiadnu novú notáciu zápisu diagramov pri návrhu, preto možno použiť notácie známe z tvorby entitno-relačných diagramov. Zavedenie označenia rôznych druhov odkazov do diagramu je prenechané na návrhára aby odzrkadľovali jeho pochopenie a potreby. HDM model podporuje tvorbu a modelovanie navigačných štruktúr, je však vhodný len pre databázovo orientované aplikácie. Informácie uložené v entitách možno znovupoužiť nielen v rámci navrhovanej aplikácie, ale aj pri tvorbe nových aplikácií. HDM sa zameriava na konceptuálny a navigačný návrh, nezaobrá sa použitím multimediálnych objektov, ani návrhom používateľských rozhraní.

3.2.2 Object Oriented Hypermedia Design Methodology (OOHDM)

OOHDM [SRB96] je objektovo orientovaná návrhová metodika založená na HDM. Využíva princípy abstrakcie a dedenia z objektovo orientovaných techník spolu s navigačnými možnosťami HDM. OOHDM sa zameriava na návrh logickej štruktúry navrhovanej aplikácie z hľadiska návrhu navigácie a rozhraní. Táto metodika napomáha pri návrhu rozsiahlych hypermediálnych aplikácií kombinovaním inkrementálneho a prototypového procesného modelu. Zahŕňa väčšiu časť životného cyklu procesu okrem ranných fáz.

Metodika je postavená na štyroch fázach: *konceptuálny návrh*, *návrh navigácie*, *abstraktný návrh rozhraní* a *implementácia*. Každá fáza umožňuje objektovo-orientovaný návrh, pričom fázy môžu byť vykonané jednotlivo, čím sa dosiahne modulárny a znovupoužiteľný návrh.

Konceptuálny návrh - V tejto fáze sa navrhuje štruktúra celej aplikácie. Na opis sa používajú objektové vlastnosti ako konceptuálne triedy, generalizácia, agregácia, inštancie tried a pod., pričom sú rozšírené o princípy odvodené z HDM a podobné perspektívam v HDM.

Návrh navigácie - V tejto fáze sa navrhujú všetky navigačné štruktúry aplikácie pre všetky možné požiadavky používateľov. Návrh navigácie môže byť zdedený z konceptuálneho návrhu, alebo môže byť vytvorený nezávisle. Na vytvorenie navigácie využíva uzly (konceptuálne triedy), odkazy, indexy a pod.

Abstraktný návrh rozhraní - V tomto kroku sa navrhuje abstraktné rozhranie spolu s rôznymi multimediálnymi typmi. Navigačné objekty z predchádzajúceho kroku sa prepájajú s objektami rozhrania. Ich rôznymi prepojeniami môžu vzniknúť rôzne rozhrania pre tie isté navigačné schémy. V tomto kroku sa tiež zdefinujú reakcie na interné a používateľom generované udalosti, synchronizácia medzi jednotlivými multimediálnymi typmi rozhrania a komunikácia medzi objektami rozhrania a navigácie.

Implementácia - V tejto časti sa výsledok návrhu navigácie a návrhu rozhraní implementuje pomocou dnes bežných nástrojov ako napr. Director, alebo pomocou jazyka HTML a pod.

Pri metodike OOHDM možno použiť niektorú štandardnú notáciu zápisu diagramov, napr. notáciu UML. Podpora multimédií v OOHDM je zabezpečená objektovo orientovanou multimediálnou databázou, ktorá zahŕňa jednak statické objekty (napr. text a obraz), ale tiež aj dynamické objekty (napr. zvuk a video) Použitie objektovo orientovanej databázy umožňuje tvorbu komplexných hypermediálnych

štruktúr pre modelovanie zložitých objektov reálneho sveta. Podobne ako HDM model, OOHDM umožňuje znovupoužitie na rôznych úrovniach návrhu. Neskoršie zmeny v návrhu pri použití OOHDM metodiky možno ľahko vykonať, avšak zmena v navrhutej štruktúre môže mať za následok nutnosť úpravy celej štruktúry. OOHDM metodika zlepšuje udržiavateľnosť a znovupoužitie vďaka oddeleniu návrhových krokov a abstrakcii objektovo orientovaného návrhu. Rozširuje HDM model o modelovacie prvky pre návrh navigácie a rozhrania. Taktiež zahŕňa interakciu medzi počítačom a používateľom.

3.2.3 The Relationship Management Methodology (RMM)

RMM [ISB95] je metodika pre štruktúrovaný návrh hypermediálnych systémov a je založená na HDM. Model pre notáciu diagramov je spojením entitno-relačného modelu na opis objektov a ich vzájomných vzťahov a HDM modelu. RMM opisuje kompletný životný cyklus projektu, avšak nezaobrá sa počítačnými fázami ako sú štúdia vhodnosti alebo analýza požiadaviek. Tiež sa nezaobrá záverečnými fázami ako je napr. údržba. Táto metodika prináša iteratívny prístup do tvorby, pretože vyžaduje spätnú väzbu medzi jednotlivými krokmi.

Samotný návrhový proces pozostáva zo siedmich krokov:

1. návrh entitno-relačného modelu - V tomto kroku sa vytvárajú entitno-relačné diagramy.
2. m-Slice¹ návrh - V tomto kroku sa modelujú atribúty entít, ktoré sa majú zobrazovať simultánne.
3. návrh navigácie - V tejto fáze návrhu sa sprístupňujú informácie pre používateľa a vytvárajú sa navigačné prvky prostredia.
4. návrh prevodu - Tu návrhár opisuje ako transformovať abstraktné konštrukcie vytvorené predchádzajúcimi krokmi do fyzickej úrovne, napr. ako ich implementovať v jazyku HTML.
5. návrh používateľských rozhraní - V tejto fáze návrhu sa rieši rozloženie textu a ostatných mediálnych prvkov.
6. návrh správania sa počas prevádzky - V tejto časti sa rieši použitie dynamicky generovaných stránok, použitie vyhľadávacích nástrojov a pod.

¹m-Slice sa používa na modelovanie malých informačných jednotiek prezentácie. m-Slice sa definuje zoskupením atribútov entity entitno-relačného modelu alebo iných m-Sliceov. Prefix *m* pochádza z pomenovania ruských bábič matrioška. Podobne ako tieto bábičky aj m-Slice môžu byť navzájom do seba poschovávané.

7. implementácia a testovanie

Pri RMM metodike možno použiť štandardnú notáciu zápisu entitno-relačných modelov, ktorá je doplnená o entity RMDM notácie. Keďže RMM je založené na princípoch návrhu databáz, je vhodné pre použitie v databázovo orientovaných web aplikáciách. Znovupoužitelnosť vytvoreného návrhu je obmedzená len na riešenia v príbuznej aplikačnej oblasti. Dodatočné úpravy v zmysle doplnenia alebo zmeny obsahu sú jednoduché, avšak úpravy štruktúry si vyžadujú väčšie zásahy do existujúceho návrhu. RMM necháva návrhárovi priestor pre návrh smerom zdola nahor ale aj zhora nadol.

3.3 Návrhové vzory

Po vzniku a rozšírení návrhových vzorov [AISJ97] v softvérovom inžinierstve sa začali návrhové vzory vytvárať aj pre web aplikácie. Možno povedať, že návrhové vzory neboli vymyslené, ale objavené v existujúcich softvérových systémoch. Návrhové vzory systematicky pomenúvajú, vysvetľujú a vyhodnocujú návrh. Zjednodušujú znovupoužitie úspešného návrhu a architektúry. Návrhové vzory opisujú problémy, ktoré sa opakovane vyskytujú, pričom zahŕňajú a vystihujú podstatu ich riešenia. Možno ich v podstate označiť za šablóny, ktoré možno použiť v rôznych situáciách. Opisujú kontext, v ktorom ich možno aplikovať, problém a interakcie, ktoré ho oživujú a elementy, z ktorých pozostáva.

Hlavné výhody použitia návrhových vzorov možno zhrnúť:

- umožňujú rozsiahle znovupoužitie softvérových architektúr
- zlepšujú komunikáciu vo vývojových tímoch vďaka zavedeniu vlastného slovníka
- zachytávajú vedomosti návrhárov, ktoré aplikujú implicitne
- umožňujú vyjadriť kompromisy a alternatívy

Spôsob zápisu návrhových vzorov pre hypermediálne aplikácie a systémy je najčastejšie vo forme textového opisu. Štruktúra opisu nie je pevná, zvyčajne však obsahuje spoločné časti. Jeden zo spôsobov zápisu je tzv. „GOF“ formát [GHJV95], ktorý obsahuje tieto časti: *názov*, *problém*, *riešenie* a *dôsledky*. Iný častý formát zápisu je tzv. „Alexanderský“ [AISJ97], ktorý obsahuje podobné časti: *názov*, *kontext*, *problém*, *riešenie* a *súvisiace vzory*. Navyše často možno nájsť aj ďalšie časti ako *motivácia*, *implementácia* alebo *príklady reálnych systémov*.

Návrhové vzory pre použitie pri tvorbe hypermediálnych aplikácií možno rozdeliť do dvoch základných skupín [RSG97]. Sú to návrhové vzory pre:

hypermediálne systémy - Do tejto kategórie zaraďujeme návrhové vzory, ktoré sa používajú pri tvorbe nástrojov a prostredí pre tvorbu hypermediálnych aplikácií. Často sa opisujú „GOF“ formátom.

hypermediálne aplikácie - Tieto návrhové vzory sú určené pre návrhárov aplikácií. Majú abstraktnejší charakter ako predchádzajúce, preto sú nezávislé od implementačných systémov. Najčastejšie sa opisujú „Alexanderským“ formátom. Návrhové vzory tejto kategórie možno ďalej rozčleniť na návrhové vzory pre:

- návrh navigácie - slúžia na organizovanie navigačnej štruktúry aplikácie v zrozumiteľnej forme
- návrh rozhraní - poskytujú návod pre rozumné organizovanie zobrazovaných informácií

Dnes sa návrhové vzory pre hypermediálne aplikácie rátať na desiatky [Hyp]. Ich kompletný opis je nad rámec tejto publikácie, preto by sme chceli poukázať na niektoré problémy z rôznych oblastí, ktoré sa snažia riešiť. Priblížime problémy a motivácie podobne, ako sú zapísané v návrhových vzoroch.

V oblasti návrhu navigácie návrhové vzory riešia napríklad tieto problémy:

- ako určiť rozsah informačného uzla ² [LRS98], [GRS97] - ako efektívne členiť množstvo informácií do samostatných celkov
- statické/dynamické informačné uzly a odkazy [GRS97] - treba riešiť pri použití web aplikácií postavených na databázovom základe
- navigačný kontext [LRS98], [GRS97] - ako organizovať navigačnú štruktúru aplikácie tak, aby sa používateľ mohol dostať k požadovanej informácii viacerými cestami
- ako poskytnúť používateľovi stálu a zrozumiteľnú informáciu o jeho pozícii v navigačnej štruktúre [GRS97] - treba riešiť nielen to, kde sa používateľ nachádza, ale aj ako mu pomôcť pri rozhodovaní kam sa chce dostať ďalej
- ako zabrániť zbytočnému „skákaniu“ používateľa medzi aktuálnym a predchádzajúcimi informačnými uzlami [LRS98] - problém pri spätnom odkazovaní sa na predchádzajúce informácie
- tvorba uzavretej navigácie v množine informačných uzlov [RSL98] - vhodné pre tematicky ucelené množiny informačných uzlov

²Pod pojmom informačný uzol môžeme rozumieť napríklad web stránku alebo stránku multimedialnej aplikácie.

- ako informovať používateľa o nových pridaných objektoch [RSL98]
- automatizované informovanie používateľa [RLS00] - poskytnúť používateľovi možnosť prihlásiť sa na automatizované zasielanie informácií
- zhromažďovanie rozhodnutí používateľa [RSL98] - vhodné pre elektronické obchody pre zaznamenanie rozhodnutí používateľa o kúpe predmetov, finančné transakcie sa ponechajú až na záver

Vzory pre návrh navigácie ďalej riešia napríklad problém ako zlepšiť navigáciu oddelením nezávislých častí web aplikácie, poskytnutím odkazov a rád [RLS00], riešia problematiku vyhľadávacích mechanizmov [LRS99] alebo personalizovaných [RSD01] a adaptívnych web aplikácií [RK02].

V oblasti návrhu rozhraní návrhové vzory riešia napríklad tieto problémy [GRS97], [LRS98]:

- ako odlíšiť obsah od navigačných prvkov používateľského rozhrania
- ako odlíšiť rôzne navigačné prvky používateľského rozhrania
- poskytnúť informácie používateľovi až na jeho žiadosť
- ako informovať používateľa o dôsledku ním aktivovaného prvku navigačného rozhrania
- ako informovať používateľa o stave prebiehajúcej interakcie

Ďalšie návrhové vzory pre návrh navigácie možno nájsť v [RSL00]. Návrhové vzory pre hypermediálne aplikácie sú opísané veľmi abstraktne a opisujú možné riešenie. Preto v nich nemožno hľadať presné postupy, skôr ich možno charakterizovať ako rozumné prístupy pre riešenie opakujúcich sa problémov. Niektorými praktickými radami sa zaoberá kapitola 4.

Kapitola 4

Návrh rozhraní

Nielen informačný obsah, ale i forma jeho prezentácie, zaujme návštevníkov web stránok surfujúcich po Internete. Atraktivnosť, zrozumiteľnosť alebo konzistentnosť internetových web sídiel môže výrazne vplývať na ich návštevnosť resp. úspešnosť. Uvedený fakt nás jasne motivuje k tomu, aby sme dostatočne dbali i na návrh web stránok z hľadiska používateľských rozhraní. Návrh používateľských rozhraní predstavuje dnes už rozsiahlu vednú oblasť, ktorá spája mnohé princípy grafického návrhárstva, psychológie a prirodzenej logiky.

Táto kapitola sa preto venuje základným aspektom návrhu web rozhraní, pričom sa zameriava najmä na internetové stránky prezentačného charakteru. Na začiatku predstavuje elementárne princípy návrhu, ktoré sú orientované na všeobecné potreby používateľa. Ďalej sú uvedené základné typy štruktúr web sídiel a opisy jednotlivých typov web stránok. V závere sú prezentované základné princípy typografie a redakčného štýlu pre optimálnu prezentáciu informačného obsahu.

4.1 Základy používateľského rozhrania

Grafické používateľské rozhrania sú navrhované na to, aby prenechali používateľovi kontrolu riadenia nad svojim počítačom. Používatelia vo všeobecnosti očakávajú od rozhraní istú úroveň sofistikovanosti, a to platí i pre web rozhrania. Hlavným cieľom vytvárania rozhraní je poskytnúť podporu potrebám používateľa, adaptovať existujúce technológie jeho požiadavkám a nikdy nevyžadovať od používateľa prispôsobenie sa obmedzeniam v ich činnosti.

Práve kvôli uvedeným aspektom web rozhraní je nutné mať základné informácie o cieľovej skupine používateľov. Je nemožné vytvoriť vhodné rozhranie pre používateľa, ktorého potreby nepoznáme. Dobrou pomôckou pri vytváraní rozhraní sú jednoduché scenáre použitia pre používateľov rôznych úrovní. Testovanie

rozhraní a spätná odozva od používateľov je najlepší spôsob ako zistiť či navrhnuté riešenia zodpovedajú potrebám návštevníkov.

4.1.1 Jasné navigačné pravidlá

Väčšinu používateľských interakcií s rozhraním web stránok tvorí navigácia cez hypertextové odkazy. Hlavným problémom je pri tom nedostatok zmysluplných informácií o tom, kde sa práve nachádzame. Jasné, jednoznačné a konzistentné ikony, grafické alebo textové mapy stránok môžu používateľovi poskytnúť dostatočnú pomoc pri nájdení toho čo hľadá, bez zbytočného strácania času. Návštevníci by mali mať možnosť jednoducho sa dostať na hlavné stránky a iné hlavné navigačné body. Tieto základné odkazy by mali byť pevnou súčasťou každej stránky. Dobrým riešením pre systém navigácie je vytvorenie konzistentného navigačného menu tlačidiel (ponúkajúce veľa možností na malom priestore).

4.1.2 Žiadne konečné stránky

Postupným prehliadaním stránok jednotlivých sekcií či podsekcii môže používateľ stratiť prehľad o tom, kde sa práve nachádza. Môže sa i stať, že neuvidí základné informácie o stránke a nebude vedieť ako sa dá dostať do jej úvodnej časti. Preto by žiadna stránka nemala byť konečná. Mala by obsahovať minimálne odkaz na návrat na hlavné stránky.

4.1.3 Efektívny prístup

Používatelia sa chcú dostať k hľadaným informáciám najmenším počtom krokov. Z toho vyplýva, že je dôležité navrhnuť efektívnu hierarchickú štruktúru informácií. Štúdie ukazujú, že používatelia preferujú menu s minimálne 5 až 7 prvkami a majú radšej menu s menším množstvom robustnejších prvkov, ako menu zložené z veľa jednoduchých prvkov.

4.1.4 Dĺžka odozvy a interakcia

Používatelia nebudú tolerovať pomalé načítavanie stránok. Prieskumy ukazujú, že pre väčšinu operácií je hranica frustrácie spôsobenej čakaním okolo 10 sekúnd. Ak je priemerný návštevník našich stránok pripojený na Internet cez modem, je nevhodné dávať na stránky rozsiahle bitmapové obrázky, pretože ich načítanie je väčšinou pomalé. Avšak, ak napríklad vyvíjame lokálnu intranetovú stránku pre viacerých používateľov, ktorí budú pripojení na web servery cez lokálnu sieť, môžeme kľudne použiť i rozsiahlejšie obrázky, animácie či multimédiá.

Vo všeobecnosti by však tvorcovia stránok mali byť konzervatívni pri používaní grafických elementov, pretože i používateľ pripojený cez vysokorýchlostný Internet (ISDN, DSL atď.) ocení aspekt rýchleho načítania.

4.1.5 Jednoduchosť a konzistencia

Návštevníci web stránok nebývajú očarení rozsiahlosťou a spleťnosťou rozhraní. A to najmä tí, ktorí potrebujú pravidelne pristupovať na dané stránky. Základnými vlastnosťami dobrých rozhraní sú jednoduchosť, zrozumiteľnosť a logickosť. Používateľské rozhranie web stránky by malo zodpovedať všeobecným navigačným a vzhľadovým konvenciám ostatnej väčšiny stránok, pretože používatelia sú už zvyknutí na tieto konvencie. Väčšinu svojho času trávia na iných stránkach ako na našej, takže je dobré vyhýbať sa veľmi nezvyčajným rozhraniam, ktoré mali pôvodne upútať pozornosť návštevníka. Pre maximálnu funkčnosť a čitateľnosť by mala byť navrhnutá stránka postavená na konzistentných vzoroch jednotlivých modulov, ktoré zdieľajú rovnaké rozloženie mriežky, rovnaké grafické témy, textové konvencie a hierarchiu. Cieľom je byť konzistentný a predvídateľný, aby používatelia mohli prezerat stránky pohodlne a mohli si byť istí, že nájdu to čo potrebujú.

4.1.6 Integrita a stabilita návrhu

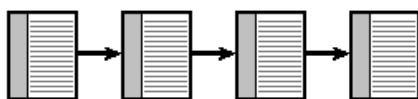
Udržiavanie funkčnej stability znamená dbať o to, aby všetky interaktívne elementy fungovali spoľahlivo. Ak navrhne a sprístupníme naše stránky, budeme musieť pravidelne kontrolovať aktivnosť všetkých odkazov a správnosť všetkých operácií.

4.1.7 Spätná odozva a dialóg

Dobre navrhnuté web stránky poskytujú priame linky na svojich správcov, prípadne kontaktné informácie na svojich zakladateľov či prevádzkovateľov. Je dôležité byť dobre pripravený na otázky a komentáre návštevníkov. Zriadenie tohto vzťahu s návštevníkmi je dobrým predpokladom na úspešné prevádzkovanie a vylepšovanie stránok.

4.2 Návrh štruktúry web sídla

Základným organizačným princípom je navrhnuť stránky v súlade s potrebami používateľov. Spýtajme sa sami seba, čo naši návštevníci chcú a zamerajme návrh stránky hlavne na tieto potreby. Veľa organizácií a obchodných spoločností robí chybu napríklad v tom, že prezentujú v prvom rade svoju organizačnú štruktúru a nie služby či produkty pre svojich zákazníkov. Zákazníka skôr zaujíma to, čo



Obrázok 4.1: Sekvenčná štruktúra

firma ponúka, než to ako je organizovaná. Rozprávajte sa s ľuďmi, ktorí tvoria vaše hlavné publikom a snažte sa vcítiť do ich predstáv.

4.2.1 Organizovanie publikovaných informácií

Vytvorená stránka nebude úspešná, pokiaľ jej obsah nebude presne, správne, atraktívne a dobre napísaný. Kognitívni psychológovia zistili, že väčšina ľudí si dokáže v krátkodobej pamäti udržať iba 4 až 7 informácií. Spôsob ako ľudia vyhľadajú a používajú informácie taktiež naznačuje, že je lepšie používať menšie jednotky informácií ako tie rozsiahlejšie. Nasledujúce štyri kroky optimalizujú organizovanie publikovaných informácií:

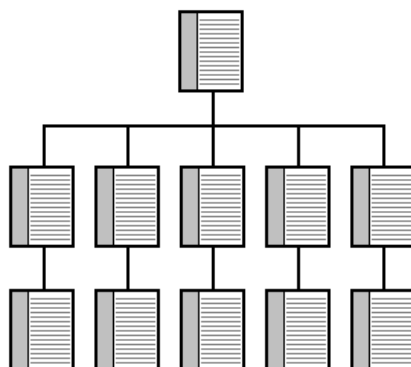
1. Rozdelíme obsah do ucelených logických častí
2. Zriadieme hierarchiu dôležitosti medzi jednotlivými časťami
3. Použijeme túto hierarchiu pri štruktúrovaní jednotlivých častí
4. Analyzujeme funkčný a umelecký úspech nášho systému.

4.2.2 Základné štruktúry stránok

Web stránky sú vytvárané podľa základných štruktúrnych architektúr. Tieto základné architektúry definujú navigačné rozhranie a pomáhajú dotvárať používateľovu mienku o tom ako sú informácie organizované. Existujú tri základné štruktúry: sekvenčná, hierarchická a sieťová.

Sekvenčná štruktúra

Najjednoduchší spôsob ako organizovať informácie je zaradiť ich do sekvenčnej postupnosti. Sekvenčné usporiadanie môže byť chronologické, alfabetické alebo logické. Sekvenčné štruktúrovanie stránok je často používané napríklad pri stránkach typu výukový kurz, v ktorých čitateľ musí prejsť postupne cez konečnú množinu stránok, ktoré využívajú lineárnu navigáciu.



Obrázok 4.2: Hierarchická štruktúra

Komplexnejšie web stránky môžu byť tiež organizované ako logické postupnosti, pričom každá stránka v postupnosti má jedno alebo viac odbočení a doplňujúcich informácií.

Hierarchická štruktúra

Hierarchická štruktúra pozostáva z jednej hlavnej stránky, okolo ktorej sú organizované ostatné podstránky. Často sa využíva pri stránkach firiem alebo organizácií.

Sieťová štruktúra

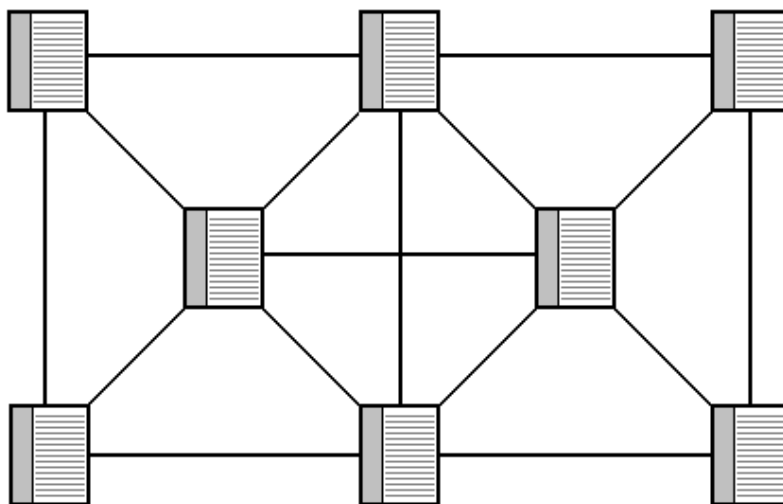
Sieťovo organizované štruktúry majú veľmi málo obmedzení. Podstatou tejto štruktúry je napodobňovanie asociatívneho myslenia človeka a veľkého toku ideí, umožňujúc používateľom voľne sledovať ich záujmy v jedinečnom, objavnom a výstrednom prostredí.

4.3 Typy web stránok

4.3.1 Hlavné stránky

Web sídla sú zvyčajne organizované okolo jednej hlavnej stránky, ktorá je vstupným bodom do systému týchto stránok. V hierarchickej štruktúre hlavná stránka je na vrchole a všetky ostatné stránky by mali obsahovať priamy odkaz na ňu. Doménová adresa (URL) našich web sídiel ukazuje väčšinou na hlavnú stránku.

Hlavné stránky môžu vykonávať množstvo funkcií. Sú najviac navštevované a teda sú veľmi vhodné pre umiestnenie noviniek, hlavného menu alebo obsahových



Obrázok 4.3: Sieťová štruktúra

osnov.

Hlavné stránky orientované na menu

Hlavné stránky orientované na menu dominovali najmä v prvých rokoch Internetu a zostali dodnes zaužívaným typom. Môžu byť navrhnuté ako zoznam textových HTML odkazov alebo obrázkové mapy s príslušnými linkami. Sofistikované návrhy využívajú kombináciu obrázkových klikacích máp a blokov textových liniek. Textové linky majú menší vizuálny efekt, avšak sa ľahšie upravujú.

Hlavné stránky orientované na nové správy

Hlavné stránky spravodajských organizácií ako New York Times a CNN sú typickými príkladmi tohto typu hlavných stránok. Aktuálne informácie robia stránky viacej zaujímavé a zvyšujú pravdepodobnosť opakovaných návštev.

Hlavné stránky orientované na prehľad poskytovaných informácií

Rozsiahle web stránky poskytujú veľké množstvo informácií pre široké spektrum návštevníkov. Navyše, návštevníci často prichádzajú na stránky so špecifickými záujmami. V tomto prípade je často výhodné rozdeliť publikum okamžite do

jednotlivých záujmových oblastí a ponúknuť im špecifickejšie informácie až cez menu podstránok.

4.3.2 Menu a podstránky

Komplexné stránky je lepšie rozčleniť do niekoľkých podstránok s vlastným menu, pretože nie je praktické a efektívne umiestniť tucty odkazov na jednu hlavnú stránku. Poskytnutím podstránky pre každý hlavný bod obsahu vytvoríme lokálne hlavné stránky pre každú sekciu.

4.3.3 Príbuzné stránky

World Wide Web rastie rapídne rýchlo a i veľké komerčné vyhľadávače ako Yahoo a Excite obsahujú linky len zlomku existujúcich stránok prístupných cez web. Keď autori začnú tvoriť web stránky, zvyčajne jednou z prvých navrhnutých stránok je zoznam stránok príbuzných ich profesii alebo záujmom.

4.3.4 FAQ stránky

Často položené otázky (Frequently Asked Questions, FAQ) sú veľmi vhodné pri stránkach, ktorých cieľom je poskytovať používateľskú podporu a iné informácie. Veľa otázok od nových používateľov bolo položených i zodpovedaných už predtým. Dobre navrhnutá FAQ stránka môže pomôcť používateľovi lepšie pochopiť relevantné informácie či ponúkané služby.

4.3.5 Chybová stránka 404

Veľa používateľov Internetu je už oboznámených s stránkou „404 error, file not found“, ktorá sa zobrazí na obrazovke, ak web server nemôže nájsť požadovanú stránku. Súbor môže chýbať, pretože ho autor premiestnil alebo zmazal, alebo používateľ jednoducho zle zadal URL adresu stránky.

Väčšina štandardných chybových stránok je všeobecná, škaredá a neinformatívna. Dobre navrhnutá chybová stránka 404 by mala byť konzistentná s dizajnom ostatných stránok. Mala by poskytovať pravdepodobné vysvetlenie chyby, navrhnovať alternatívy a poskytnúť odkazy na lokálne hlavné stránky, obsah stránok či vyhľadávaciu stránku.



Obrázok 4.4: Príklad zlého a nekonzistentného návrhu (vľavo), príklad konzistentného a dobre čitateľného návrhu (vpravo) [LH02]

4.4 Typografia

Typografia sa zaoberá rovnováhou a vzájomným pôsobením typov písma na stránke, slovné a vizuálne porovnanie, ktoré pomáha používateľovi porozumieť forme a absorbovať podstatu obsahu stránky.

4.4.1 Čitateľnosť

Dobrá typografia závisí od vizuálneho kontrastu medzi použitými fontami, odstavcami, nadpismi a okolitým prázdny bielym miestom. Nič nezaujme oko a mozog čitateľa tak, ako silný kontrast a zreteľné vzory. Návrhár tento efekt môže dosiahnuť iba dôsledným návrhom týchto jednotlivých častí stránky.

Ak náš obsah tvorí hlavne text, typografia je dobrý nástroj, ktorý môžeme využiť na „nakreslenie“ vzorov organizácie textu na stránke. Prvá vec, ktorú si čitateľ všimne nie je nadpis alebo iné detaily na stránke, ale je to celkový vzor a kontrast stránky. Pravidelné opakované vzory zavedené cez dôsledné organizovanie textu a grafiky pomôže zvýšiť úroveň čitateľnosti. Naopak nejednotná heterogénna typografia sťažuje čitateľovi orientáciu v texte.

4.4.2 Zarovnanie

Okraje definujú čitateľský úsek stránky tým, že oddeľujú hlavný text od okolitého prostredia. Okraje vytvárajú dôležitý vizuálny reliéf v každom dokumente, ale dôsledný návrh okrajov a ostatného „čistého miesta“ je dôležitý najmä pri web stránkach, pretože obsah web stránok musí súčasne existovať na obrazovke počítača spolu s elementami rozhrania prehliadača a s ostatnými oknami a ikonami používateľských rozhraní. Okraje a voľné miesto môžu byť využité na zvýraznenie hlavného textu od ostatných častí. Odstavce textu môžu mať trojaké zarovnanie: zarovnanie doľava, zarovnanie doprava a zarovnanie na stred.

4.4.3 Typy fontov

Každý typ fontu má špecifický tón, ktorý by mal vytvárať harmonický súlad medzi slovným a vizuálnym tokom obsahu. V prvých verziách jazyka HTML, sa typy fontov nedali zdefinovať. Fonty sa zobrazovali tak, ako ich mal používateľ nastavené vo svojom prehliadači. Aktuálne verzie jazyka HTML a CSS umožňujú používať dizajnérom široké palety fontov, ktoré štandardne podporujú operačné systémy. Ak sa dizajnérom zdefinovaný font nenachádza na lokálnom počítači, prehliadač na príslušných stránkach zobrazí font, ktorý je nastavený ako predvolený.

4.4.4 Čitateľnosť fontu na obrazovke

Niektoré typy fontov sú viac čitateľnejšie na obrazovke ako ostatné. Tradičný typ fontu ako Times Roman sa považuje za jeden z najviac čitateľných fontov na tlačenom papieri, avšak na obrazovke počítača vyzerajú jeho písmená príliš malé a nepravidelné. Čitateľnosť fontu na obrazovke je zvyčajne najviac ovplyvnená tzv. x-výškou (výška malého písmena „x“) a celkovou veľkosťou fontu.

4.4.5 Fonty navrhnuté pre obrazovku počítača

Fonty ako Georgia a Verdana boli špeciálne navrhnuté pre optimálnu čitateľnosť na obrazovke počítača. Majú prehnanú x-výšku a sú širšie v porovnaní s ostatnými tradičnými fontami. Tieto fonty ponúkajú vynikajúcu čitateľnosť pre web stránky.

4.4.6 Výber fontu

Najkonvenčnejšia schéma fontov je použitie pätkového písma ako Times New Roman alebo Georgia pre bloky textov a bezpätkového ako Verdana či Arial ako kontrastné nadpisy. Vo všeobecnosti nastavujeme font Times New Roman na web stránkach, pretože vytvára odôvodnenú rovnováhu medzi hustotou informácií a

Arial

Každý typ fontu má špecifický tón, ktorý by mal vytvárať harmonický súlad medzi slovným a vizuálnym tokom obsahu.

Georgia

Každý typ fontu má špecifický tón, ktorý by mal vytvárať harmonický súlad medzi slovným a vizuálnym tokom obsahu.

Verdana

Každý typ fontu má špecifický tón, ktorý by mal vytvárať harmonický súlad medzi slovným a vizuálnym tokom obsahu.

Times New Roman

Každý typ fontu má špecifický tón, ktorý by mal vytvárať harmonický súlad medzi slovným a vizuálnym tokom obsahu.

Obrázok 4.5: Ukážky spomínaných fontov

celkovou čitateľnosťou. Väčšina používateľov očakáva pätkové písmo pre dlhé bloky textov a považujú Times New Roman za najvhodnejší pre tlačené dokumenty. Viaceré štúdie potvrdzujú, že pätkové písmo je čitateľnejšie oproti bezpätkovému. Dizajnér môže použiť buď variácie pätkových fontov, alebo kontrastnejšie bezpätkové fonty. Najbezpečnejšie je použiť jednoduchú typografickú rodinu fontov a meniť ich veľkosť a hrúbku písma. Taktiež môžeme zvoliť kombináciu pätkového a bezpätkového písma, ale mali by sme vyberať iba kompatibilné fonty.

4.5 Redakčný štýl

Ľudia čítajú inak na web stránkach ako na tlačných dokumentoch. Jeden z dôvodov je, že čítanie textu na obrazovke je nepohodlné. Vďaka nízkemu rozlíšeniu obrazovky počítača a ťažkopádnosti skrolovania, veľa používateľov si texty na web stránkach iba prehliadne a potom si ich vytlačí pre pozornejšie čítanie. Ďalší dôvod je ten, že používatelia sa presúvajú zo stránky na stránku, pričom zbierajú informácie z rôznych zdrojov. Chcú rýchlo zistiť obsah stránky, nájsť čo hľadajú a potom ísť ďalej.

4.5.1 Štylizovanie prózy

Dokumenty publikované na web stránkach musia byť výstižné a dobre štruktúrované kvôli efektívnejšiemu prehľadávaniu. Ľudia zvyknú skôr preskakovať po

textoch na stránke ako čítať slovo za slovom. Je vhodné použiť nadpisy, zoznamy, typografické zvýraznenia pre slová alebo sekcie, ktoré chceme zvýrazniť. Zvýraznené elementy by mali byť jasné a presné. Štýl „obrátenej pyramídy“ (záver na začiatku textu) používaný v žurnalistike funguje dobre i pri web stránkach. Odporúča sa umiestniť dôležité fakty z dokumentu do prvých odstavcov, kde si ich čitatelia môžu ľahko nájsť.

4.5.2 Nadpisy a podnadvpisy

Edičné orientačné body ako nadpisy a podnadvpisy sú základným rozlišovacím a štruktúrovacím nástrojom vo web stránkach ako i v každom inom type publikácie. Konzistentný prístup k titulom, nadpisom, podnadvpisom v dokumentoch pomôže čitateľovi lepšie sa orientovať.

4.5.3 Odporúčané textové štýly

Pre nadpisy (titul stránky, referencie na ostatné stránky, tituly stránky spomenuté v texte, mená produktov, obchodné značky) sa odporúča použiť tučné písmo, pričom začiatkové písmeno každého slova napíšeme veľkým písmenom.

Pre podnadvpisy (podnadvpisy na stránke, referencie na ostatné sekcie stránky, opisy grafov, obrázkov, zoznamy) sa odporúča použiť tučné písmo, pričom iba začiatkové písmeno prvého slova napíšeme veľkým písmenom.

4.6 Záver

Návrh rozhraní web stránok predstavuje nezanedbateľnú súčasť procesu web inžinierstva. Je súborom techník a metód pre optimálne prezentovanie informačného obsahu. Jeho fundamentálnym princípom je snaha dbať na požiadavky používateľa, a tým im čo najviac spríjemniť resp. uľahčiť surfovanie po web stránkach. V dnešnej dobe už existuje mnoho štúdií, ktoré ho podrobne skúmajú, a tie najzákladnejšie princípy boli prezentované v tejto kapitole. Rozvojom web technológií či samotnej informačnej spoločnosti sa však táto oblasť neustále vyvíja, a preto je vhodné pravidelne sledovať jej aktuálne trendy. Kvalita, atraktivnosť a zrozumiteľnosť web rozhraní bude totiž vždy významne vplývať na návštevnosť resp. úspešnosť celých web projektov.

Kapitola 5

Testovanie a nasadenie

V tejto kapitole bližšie opisujeme testovanie web aplikácií. Uvádžeme niektoré bežné postupy a metódy používané pri testovaní softvérových aplikácií a systémov a ich porovnanie s web testovaním. Ďalej sa tu zaoberáme oblasťami web aplikácií, na ktoré sa testy hlavne zameriavajú a v krátkosti opisujeme hierarchiu a organizáciu vykonávaných testov so zameraním na internetové aplikácie typu *klient-server*.

5.1 Prečo testovať

Ideálnym cieľom vývoja softvéru je dosiahnuť nulovú chybovosť produktu. Prax je však iná - hovorí sa, že v každom softvéri je minimálne jedna chyba. Napriek tomu, že programátori často považujú testovanie za zdĺhavú a o čas oberajúcu činnosť, v softvérovom procese by malo mať svoje pevné miesto. Dobre navrhnuté testy môžu odhaliť veľké množstvo chýb, ktoré sa následne dajú redukovať na minimálnu úroveň. V nasledujúcich bodoch uvedieme niekoľko ďalších dôvodov, prečo je dôležité softvér testovať:

- **Overenie, že všetky požiadavky boli analyzované správne**

Veľa zo závažných zlyhaní softvéru bolo zapríčinených nekompletnými alebo chýbajúcimi požiadavkami. Testovanie overuje, či sú požiadavky relevantné, koherentné, merateľné, kompletne a testovateľné.

- **Overenie, že všetky požiadavky boli implementované správne**

Adekvátne testovanie zabezpečuje, že softvér pracuje tak, ako sa očakáva. Na jednotlivé dopyty používateľa odpovedá správne a pracuje podľa zadanej špecifikácie.

- **Identifikovanie chýb v skorších fázach vývoja produktu**

Čím skôr sa chyba v systéme odhalí, tým menej stojí jej odstránenie.

- **Verifikovanie interakcií medzi softvérovými súčiastkami**

Množstvo chýb v systémoch vzniká práve pri interagovaní častí systému. Vhodným návrhom testov môžeme tieto chyby identifikovať a následne odstrániť.

- **Zvyšovanie spoľahlivosti aplikácie**

Testovaním sa chyby v systéme odhaľujú¹. Následne môžu byť opravené, čo prispieva k zvyšovaniu spoľahlivosti produktu.

5.2 Testovanie vo všeobecnosti

Testovanie zastáva dôležitú úlohu v procese vývoja softvéru. Môžeme ho nájsť prakticky vo všetkých fázach vývoja. Od plánovania akceptačných testov (špecifikácia požiadaviek), plánovania systémových integračných testov (špecifikácia systému), plánu testovania jednotiek (hrubý a jemný návrh), regresného testovania počas implementácie až po akceptačné testovanie zákazníkom.

Vzhľadom na fakt, že pojem testovanie aplikácií v sebe zahŕňa širokú škálu pojmov, uvádzame krátky prehľad niektorých spôsobov testovania [sal].

Testovanie kompatibility - testovanie pre zaistenie kompatibility aplikácie alebo web stránky na rôznych prehliadačoch, operačných systémoch a hardvérových platformách. Môže sa vykonávať manuálne alebo automatizovane, použitím testovacích sád.

Testovanie zhodnosti - overenie zhodnosti vzhľadom na výrobné štandardy (prenosnosť, prevádzkyschopnosť, kompatibilitu s definovanými štandardmi).

Funkcionálne testovanie - overenie, či aplikácia spĺňa svoju špecifikáciu a správne vykonáva všetky svoje funkcie.

Testovanie nasadenia - generické testovanie, ktoré v sebe zahŕňa *prevádzkové testovanie* a *záťažové testovanie*.

Prevádzkové testovanie - overenie zhody systémových komponentov vzhľadom na prevádzkové požiadavky. Často sa používajú automatizované testovacie nástroje na simuláciu veľkého počtu používateľov.

¹napr. odhalenie nepredpokladaného správania sa aplikácie

Regresné testovanie - spätné, opakované testovanie komponentov za účelom overenia funkčnosti. Využíva sa hlavne pre overenie vplyvu nových zmien v systéme na doterajšiu funkčnosť ostatných častí. Môže sa vykonávať manuálne, ale často sa používajú testovacie prípady a sady pre urýchlenie opakovaných úkonov.

Dymové testovanie - hrubé testovanie hlavných funkcií systému bez kladenia dôrazu na detaily.

Zátťažové testovanie - sledovanie systému za hranicami jeho funkčnej špecifikácie. Často sa vykonáva podobným spôsobom ako *prevádzkové testovanie*, ale s použitím omnoho vyššej úrovne testovacej prevádzky.

Testovanie jednotiek - funkcionálne a zhodové testovanie samostatných systémových jednotiek.

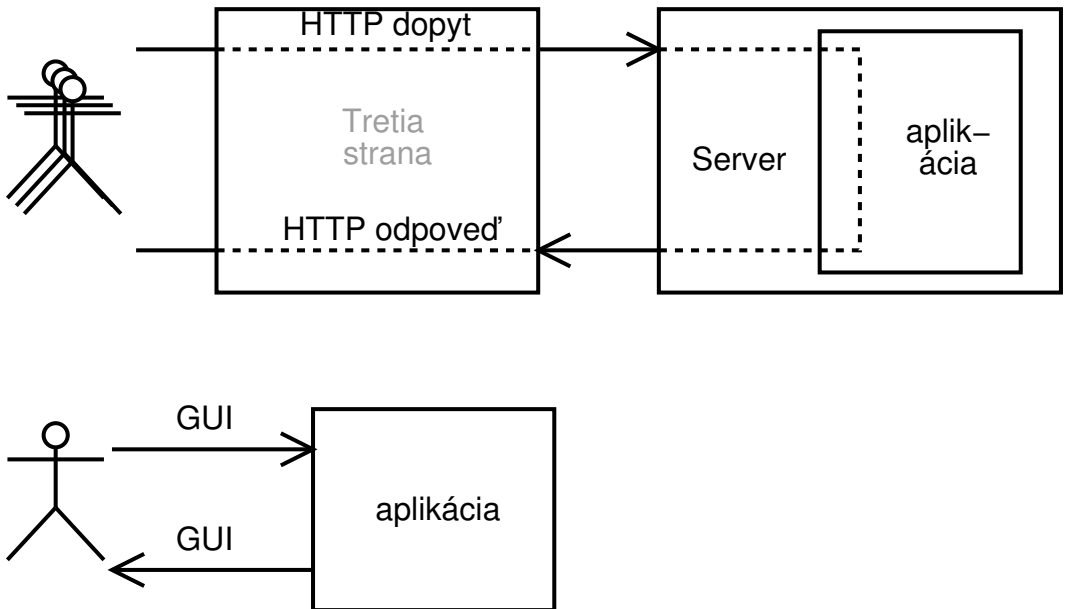
5.3 Rozdiely medzi testovaním web aplikácií a klasických softvérových aplikácií

S prudkým rozvojom Internetu vo svete sa počet systémov, ktoré sú založené na sieťových technológiách stále zvyšuje. Spolu s množstvom web aplikácií sa tak tiež zvyšuje aj ich komplexnosť a zložitnosť, čo samozrejme vedie k väčšiemu počtu chýb. Preto je potrebné vyvinúť väčšie úsilie pri testovaní a hlavne testovanie zautomatizovať. Existuje veľa postupov, metód a nástrojov pre testovanie klasických softvérových aplikácií. My sa budeme zaoberať rozdielmi medzi testovaním web aplikácií a testovaním klasických softvérových interaktívnych aplikácií.

Môžeme povedať, že väčšina web alebo klasických softvérových aplikácií, sa správa ako „oknová aplikácia“². Všetky komunikujú cez používateľské rozhranie s jedným alebo viacerými používateľmi. Napriek tomu medzi nimi môžeme nájsť zásadnejšie rozdiely. Web aplikácie, na ktoré sa hlavne zameriame my, sú založené na komunikácii *klient-server*. Aplikácia sa vykonáva na serveri, ktorý je pripojený na Internet. Cez toto pripojenie, môžu klienti komunikovať s aplikáciou použitím predpísaných protokolov. Na obrázku 5.1 je zobrazený schematický náčrt komunikácie internetovej a klasickej softvérovej aplikácie. [vBM03]

Pri klasických softvérových aplikáciách používateľ komunikuje len s grafickým rozhraním (GUI), kým pri internetových aplikáciách požiadavka klienta prechádza tretou stranou, web serverom až k aplikácii samotnej. Ako môžeme vidieť na obrázku 5.1, pri testovaní web aplikácií sa musíme sústrediť na päť entít: [vBM03]

²vybrali sme tento druh aplikácií vzhľadom na fakt, že sú najčastejšie používané



Obrázok 5.1: Schematické porovnanie komunikácie a) Internetovej aplikácie b) klasickej softvérovej aplikácie

1. klient
2. komunikačné protokoly
3. tretie strany
4. web server
5. aplikácia samotná

Klient

Používateľské strany - klienti, na ktoré sa v tejto kapitole budeme sústreďovať my, sa nazývajú *tenkí klienti*³. Sú to prevažne web prehliadače. Na rozdiel od klasických softvérových aplikácií, môže pristupovať k internetovej aplikácii súbežne viacero klientov. Klient môže zlyhať, alebo skončiť a to bez porušenia chodu celej aplikácie.

³Majú redukované (alebo žiadne) výpočtové možnosti. Všetka aplikačná logika je na strane servera. *Tučný klient* - aplikačná logika je implementovaná čiastočne aj na strane klienta.

Tretia strana

Pretože komunikácia vo web aplikáciách prebieha cez Internet, sú závislé od tretej strany. Pakety sú posielané cez smerovače, ktoré riadia komunikáciu na sieti. Nemôžeme ovplyvniť, cez ktoré smerovače a uzly dáta pôjdu. Okrem toho, komunikáciu ovplyvňujú aj ďalšie okolnosti - preklad adres cez DNS server, tretie strany pre verifikáciu dát a pod.

Komunikačné protokoly

Väčšina aplikácií na báze webu je založená na HTTP protokole, ktorý je typu *dopyt - odpoveď*. Server čaká na dopyt, akonáhle príde, vybaví požiadavku a pošle odpoveď. Časy odozvy však závisia od stavu siete a komunikácia môže zlyhať.

Web server

Web server je softvérový systém, ktorý je pripojený k Internetu. Na rozdiel od klasických softvérových aplikácií, klient sa môže snažiť prístupit' k web serveru, ktorý je vypnutý alebo pretiažený, čo spôsobí zlyhanie interakcie.

Aplikácia

K aplikáciám, na ktoré sa zameriavame môže naraz pristupovať viacero klientov. Niekde by teda mala byť uložená informácia o tom, kto komunikuje s aplikáciou. Operácie (dopyty a odpovede) určené pre jedného klienta sú uložené v tzv. *session*.

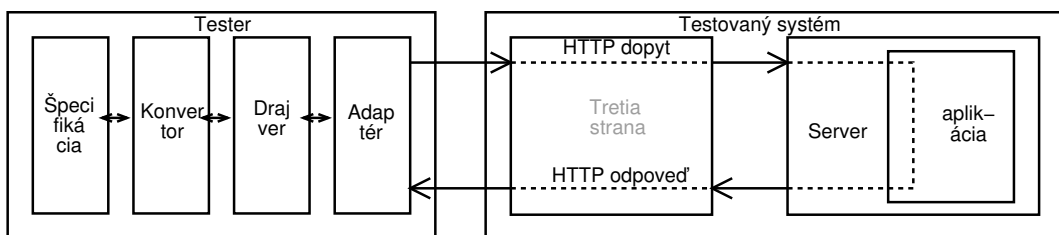
5.4 Návrhy testovania web aplikácií

Po náčrte hlavných oblastí, na ktoré sa potrebujeme zamerať pri testovaní môžeme prejsť k návrhu samotného automatizovaného testovania. Spôsobov ako navrhovať testy pre web aplikácie je veľa. Spomenieme niektoré z nich.

5.4.1 Dynamické testovanie metódou čiernej skrinky

Testovanie pozostáva z opakovaného vykonávania implementovaného systému v reálnych podmienkach. Dosiahneme to simulovaním reálnych interakcií s aplikáciou, teda používaním testovacích klientov, ktorí sa správajú tak ako klienti pri reálnom nasadení systému. Na obrázku 5.2 je schéma testera.

Tester generuje požiadavky (dopyty) a prijíma odpovede, na základe ktorých vyhodnocuje správnosť testovaného systému. Tester sa skladá zo štyroch častí:



Obrázok 5.2: Schematické navrhnutie testera pre dynamické testovanie metódou čiernej skrinky

Špecifikácia - formálny opis správania sa systému.

Konvertor - komunikuje so špecifikáciou a udržiava záznamy o testovacom stave. Navyše porovnáva očakávanú špecifikáciu s dosiahnutými výsledkami.

Drajver - centrálna jednotka, ktorá kontroluje vykonávanie testov. Určuje, ktoré testy sa majú vykonať. Vyhodnocuje úspešnosť testovania aplikácie.

Adaptér - slúži na prepis abstraktnej reprezentácie požiadaviek do HTTP požiadaviek a dekódovanie HTTP odpovedí do abstraktnej reprezentácie.

Počas spustenia testu *drajver* vyhodnocuje, či bol dopyt odoslaný alebo odpoveď prijatá. V prípade odoslania dopytu, *drajver* požiada *konvertor* o zaslanie správnej požiadavky odvodennej zo *špecifikácie*. Požiadavka je preložená *adaptérom* a poslaná aplikácii na testovanie. V prípade očakávania odpovede je táto najskôr dekódovaná *adaptérom*. *Konvertor* vyšle správnu odpoveď vychádzajúcu zo *špecifikácie* a na základe porovnania môže byť vynesený verdikt o úspešnosti alebo neúspešnosti testu.

5.4.2 Analýza a reprodukcia chýb vo web aplikáciách

Testovanie web aplikácií má veľa spoločného s testovaním klasických softvérových systémov. Okrem testovania štandardnej funkcionality, konfigurácie, kompatibility a vykonania štandardných testovacích typov je však treba otestovať aj interakcie medzi aplikáciou a klientmi. Práve táto časť je značne komplikovaná. Nie je vždy ľahké nájsť alebo zreprodukovať chybu. Pri odhaľovaní takýchto chýb je dobré dodržať nasledovné testovacie princípy: [Ngu00]

1. *Našli sme chybu alebo len symptóm?* Bez analýzy prostredia si nemôžeme byť istí, či je objavený symptóm následok chyby, alebo len zlej konfigurácie na strane klienta.

2. *Chyby môžu byť závislé od prostredia a nemusia byť vždy viditeľné .*
3. *Chyby môžu byť v kóde alebo v konfigurácii. Chyby môžu byť v kóde, alebo len v nastavení siete, servera, resp. klienta. Nie je dobré hneď robiť zmeny v kóde za účelom odstránenia chyby predtým, ako zistíme skutočnú príčinu chyby.*
4. *Chyby sa môžu nachádzať v rôznych vrstvách. Analýza pôvodu chyby z hľadiska úrovne jej vzniku dokáže odstrániť ďalšie problémy.*
5. *Statické (konfiguračné chyby a chyby kompatibility) a dynamické chyby (časovo závislé chyby a chyby zdrojov) vyžadujú rôzne postupy pri ich odhalovaní.*

5.4.3 Testovanie bezpečnosti vo web aplikáciách

Okrem vyššie spomenutých oblastí testovania internetových aplikácií je potrebné venovať zvýšenú pozornosť bezpečnosti a neautorizovaným vstupom do aplikácie. Táto časť je pomerne rozsiahla, preto sa zameriame len na niektoré používané techniky. Viac informácií je dostupných v [WT02].

Mapovanie siete - zisťovanie neautorizovaných prístupov do siete a k aplikácii.

Zisťovanie zraniteľnosti - používajú sa automatické skenery na zistenie zraniteľnosti aplikácie. Pri zistení defektu môžu problémy čiastočne odstrániť, resp. zaslať správu alebo záznam o defekte.

Testovanie prieniku - bezpečnostné testovanie s cieľom odhaliť chyby a napadnúť systém zvonku. Pre toto testovanie sa používajú techniky vyvinuté hakermi.

Bezpečnostný test a vyhodnotenie - vyhodnotenie bezpečnosti systému po jeho integrovaní a zavedení. Je to analýza a skúška bezpečnostných opatrení systému.

Napadnutie hesiel - identifikovanie slabých (jednoduchých) hesiel na základe vygenerovaných postupností.

Prehliadka záznamov - prezeranie tzv. log súborov systému pre zaistenie bezpečnosti systému (záznamy firewallu, záznamy servera a pod.)

Zistenie integrity súborov - vytváranie a následné overovanie kontrolnej sumy pre súbory, ktoré by mali byť chránené pred neautorizovaným prístupom.

Detekcia vírov - automatizované zisťovanie napadnutia vírmi, trójskymi koňmi a pod.

5.5 Techniky, metódy, podporné programy a systémy

Testovanie je činnosť, ktorá sa vo všeobecnosti nepovažuje za príjemnú časť procesu vývoja aplikácie. Na jej urýchlenie a automatizáciu bolo vytvorené veľké množstvo podporných programov a nástrojov. Skoro pre každú časť softvérového inžinierstva by sme mohli nájsť vlastné testovacie metódy a postupy. Web inžinierstvo v tejto oblasti rozhodne nezaostáva. Boli vyvinuté nástroje pre testovanie častí aplikácií od rozhrania, bezpečnosti, databázy, klientov až po jednotlivé moduly. Pre základný prehľad existujúcich nástrojov uvádzame krátku tabuľku (detailnejšie rozdelenie je v [sal]).

<i>Oblasť testovania</i>	<i>Vybrané produkty</i>
HTML testy	Free Link Checker, DeadLinks, HTML PowerTools
Java testy	Abbot, Cactus, BugKilla, JUnit
Databázové testy	Data Generator, DataRecon
API testy	Robot, DejaGNU
Manažment testov	Extended Test Plan, QADirector

Obrázok 5.3: Tabuľka vybraných produktov pre rôzne oblasti testovania v softvérovom inžinierstve

5.6 Nasadenie web aplikácií

Ak je aplikácia vytvorená a otestovaná, môžeme pristúpiť k fáze nasadenia a údržby. Fáza nasadenia úzko súvisí s testovaním systému. Ako sme spomenuli vyššie, testovanie web aplikácií je relatívne komplikovaný proces, závislý aj od konkrétnych podmienok, v akých sa aplikácia vykonáva, a klientov s ňou komunikujúcich. Pre úplný chod aplikácie treba zabezpečiť funkčnosť servera, klientov, databázy a ich vzájomné napojenie (na Internet, sieť a celkovú komunikáciu).

Fáza údržby by sa dala rozdeliť do troch hlavných častí [Gin02]:

1. Obsahová
2. Softvérová
3. Hardvérová a sieťová údržba

Údržba systému po obsahovej stránke úzko súvisí z návrhom systému. Čím lepší návrh systému, tým ľahšia údržba.

Činnosti softvérovej údržby môžu byť rozdelené do štyroch oblastí:

1. Opravná
2. Preventívna
3. Zdokonaľovacia
4. Adaptívna

Často sa v systéme po nasadení objavia rôzne chyby, ktoré je nutné odstrániť. Treba doimplementovať a odstrániť chyby a spísať zmeny v dokumentácii. Táto časť údržby sa nazýva *opravná údržba*. Taktiež sa počas údržby snažíme predísť ďalším problémom pred tým, ako sa vyskytnú. Opravovanie takéhoto druhu chýb patrí do *preventívnej údržby*. Napriek tomu, že systém funguje bez obmedzení, môžeme prísť s lepším riešením implementácie. Tento proces je známy ako *zdokonaľovacia údržba*. Ak sa požiadavky na systém zmenia (napr. zvýšenie daňovej sadzby) potrebujeme vykonať zmeny - prispôbiť systém novým požiadavkám (*adaptívna údržba*).

Tak isto potrebujeme pravidelne udržiavať a obnovovať hardvér, aby sme sa vyhli chybám, ktoré sa v súvislosti s ním vyskytnú.

Je dobré rozdeliť si údržbu do viacerých častí. Pri prijatí požiadaviek z tejto oblasti je ich potom možné zaradiť a stanoviť si priority pre dané oblasti údržby.

Pri web aplikáciách prichádza do úvahy ešte ďalší typ údržby súvisiaci s časťým prepájaním aplikácií prostredníctvom odkazov - kontrola platnosti odkazov. Pri celkovej údržbe je potrebné zaoberať sa aj touto problematikou, vzhľadom na časté zmeny v tejto oblasti.

Na záver by sme radi spomenuli, že testovanie, nasadenie a údržba web aplikácií má veľa spoločných črt s testovaním nasadením a údržbou klasických softvérových aplikácií. Je im však treba venovať zvýšenú pozornosť vzhľadom na vplyv tretej strany (siete).

Kapitola 6

Manažment

Tvorba web aplikácií je komplikovaný technický proces. Vyžaduje mnoho ľudí, často pracujúcich súčasne. Kombinácia technicky a netechnicky zameraných úloh, ktoré sú nutné na vytvorenie vysokokvalitnej web aplikácie predstavujú výzvu pre akúkoľvek skupinu profesionálov. V snahe vyhnúť sa zmätkom, neuspokojivým výsledkom a neúspechu je nutné plánovanie, zváženie rizík, zavedenie a plnenie plánu a definovanie spôsobu kontroly. Toto sú základné činnosti, ktoré nazývame manažment projektov. Teoreticky väčšinu činností, ktoré sú definované pre manažment klasických softvérových projektov môžeme aplikovať na web projekty. Ale v praxi je prístup web inžinierstva k manažmentu projektov značne odlišný.

6.1 Motivácia

Manažment projektov tvorby web aplikácií je určený jedinečnou charakteristikou týchto aplikácií. Objavujú sa otázky, ktorých odpovede môžu spôsobiť úspech aj krach projektu:

- Vývoj web aplikácií je relatívne nová aplikačná oblasť. Máme k dispozícii malé množstvo historických údajov a projektovo orientovaných metrík, ktoré môžeme použiť na odhadovanie. Ako získať spoľahlivé odhady? Aký stupeň istoty nám dá definovaný plán?
- Odhady, analýza rizík a plánovanie sú určené na základe jednoznačného porozumenia rozsahu projektu. Ale rozsah web aplikácií sa počas ich vývoja veľmi často mení. Ako môžeme kontrolovať náklady a plán, keď sa požiadavky často dramaticky menia v priebehu projektu? Ako môžeme kontrolovať zmenu rozsahu projektu? Môže byť kontrolovaný aj keď základná charakteristika web aplikácií je ich jedinečnosť?

Pri súčasnej úrovni vývoja manažmentu web projektov nie je vždy jednoduché odpovedať na tieto otázky. Napriek tomu máme niekoľko pravidiel, nad ktorými sa môžeme zamyslieť.

Plán vývoja web aplikácií je zvyčajne obmedzený relatívne krátkym časovým intervalom (často kratší ako jeden alebo dva mesiace), preto by mal mať vysoký stupeň granularity. Preto by mali byť úlohy a menšie kontrolné body naplánované s granularitou jedného dňa. Jemná granularita umožňuje zákazníčkovi aj manažérovi rozoznať sklz pred tým, než ohrozí dátum odovzdania finálneho produktu.

Faktor jedinečnosti projektu často spôsobuje, že pri odhadoch nemôžeme postupovať na základe vedomostí z iných projektov, pretože sú značne rozdielne. Iteratívny vývoj poskytuje členom vývojového tímu možnosti učenie sa a zlepšovanie projektu počas jeho trvania. Štandardný manažment projektov odporúča, aby sme spätne skúmali projekt po jeho ukončení, aby sme sa učili a zlepšovali. Pre web projekty to môže byť neskoro. Ak sa učíme po skončení projektu, predpokladáme, že ak dané podmienky nastanú znovu, urobíme veci rozdielne a aplikujeme to, čo sme zistili a budeme úspešnejší. Úspech tejto taktiky závisí od ľudí. Predpokladáme, že budú študovať zistenia vyplývajúce z predchádzajúcich projektov pred začatím nových. Tiež dúfame, že sa projekty uskutočnia na podobných technologických a obchodných doménach a tímy budú dynamické. Všetko, čo sa naučíme, môžeme aplikovať - čo je zvyčajne dost' zriedkavé.

Dobré iteratívne projekty pozostávajú z malých častí, na ktorých sa učíme a spätne skúmame výsledky s našimi predpokladmi na konci každej iterácie. Tímu položíme otázky: „Čo sa podarilo, čo išlo podľa plánu?“, „Čo nie?“, „Aké sú odporúčania pre ďalšiu iteráciu?“ Vzhľadom na krátky časový rámec menej vecí zabudneme. Tím sa učí počas projektu a získané skúsenosti priamo používa v ďalšej iterácii. To mu umožní stať sa efektívnejším, vyprodukovať vyššiu hodnotu a zlepšiť kvalitu. Výsledky a zistenia predchádzajúcich iterácií sú plne aplikovateľné v ďalších, pretože obchodná a technologická doména je rovnaká a nemenia sa osoby, ktoré sú do projektu zapojené.

V projektoch, ktoré nevyvíjame iteratívne, môže spôsobiť nezvládnutie jednej etapy neúspech celého projektu. Nezvládnutie jednej etapy pri iteratívnom vývoji spôsobí, že sa poučíme, adaptujeme sa a zlepšime sa v ďalšej novej iterácii. V tomto prípade je riziko redukované a množstvo výsledkov sa zvýši.

Rozsah projektu sa bude meniť tým, ako bude web projekt postupovať, preto by mal byť web inžiniersky proces inkrementálny. Toto umožní tímu pevne určiť rozsah pre jeden inkrement, takže môže vytvoriť prevádzkyschopný produkt. Ďalší inkrement môže odrážať zmeny rozsahu navrhnuté na základe predchádzajúceho inkrementu, ale akonáhle sa zahájí ďalší inkrement, rozsah je znovu dočasne pevne určený. Takýto postup umožní tímu pracovať bez nutnosti byť vystavený nepretržitému prúdu zmien, a aplikácia sa stále priebežne vyvíja.

6.2 Plánovanie

Spôsob, akým vyvíjame softvér sa mení. Skôr než vývoj softvéru z požiadaviek cez vodopádový model, vývoj web aplikácií prebieha spájaním stavebných blokov a znovupoužiteľných komponentov použitím rýchlych metód vývoja aplikácií a neustálym prototypovaním. Manažment takýchto projektov môže počas vývoja spôsobovať nočné mory. Veci sa dejú tak rýchlo, že je ťažké mať pod kontrolou ich stav a to, či primerane napredujú. Postup vývoja web aplikácií je tiež ťažké odhadnúť. Zvlášť v prostredí s limitovanými zdrojmi potrebujeme pre vývoj softvéru lepšiu predpoveď času a úsilia vyžadovaného na úspešný priebeh takýchto projektov. Realistické odhady a plánovanie vyžadovaného úsilia v životnom cykle web aplikácie umožnia manažérom a vývojárom efektívne plánovať zdroje.

6.2.1 Plán projektu

Plánovanie je proces identifikácie práce, ktorá musí byť vykonaná a spôsobu ako bude dokončená v rámci schválených časových a zdrojových obmedzení. Kľúčovým výstupom z procesu plánovania je plán projektu, ktorý špecifikuje trvanie aktivít a čas ich začatia. Naším cieľom je vytvoriť flexibilný dokument, ktorý bude schopný zachytávať zmeny. Z perspektívy nášho projektového tímu plán poskytnete odpovede na niekoľko kľúčových otázok:

- čo úloha vyžaduje od plníteľa
- čo musí byť vykonané predtým, než začne
- kedy musí byť ukončená
- aká úloha nasleduje

Vieme, že plán rýchlo zastará v dôsledku toho, že požiadavky klienta menia prioritu a pribúdajú. Sponzori menia obchodné priority po celú dobu trvania a napredovania projektov a ich vízie toho, ako sa koncový produkt kryštalizuje. Dá sa vôbec ubrániť myšlienke, že s takýmito skutočnosťami je plánovanie stratené? Pokúsme sa o to. Prvým krokom je poznanie, že plánovanie je umenie zamaskované do vedy. Plán projektu by mal byť dynamický, žijúci dokument, navrhnutý tak, aby bol čo najviac flexibilný. Jeho zámerom je definovať hlavné toky a závislosti medzi jednotlivými činnosťami projektu. Mal by byť definovaný tak, aby vyhovoval jedinečnej charakteristike projektu. Pre jednoduché projekty nemá zmysel snažiť sa prispôbovať ich plány do všeobsahujúcich šablón, ktoré sa pokúšajú zachytiť a stanoviť detail na každú minútu. Vývojoví špecialisti si vedia naplánovať ich vlastný čas. Členovia tímu potrebujú plán, aby vedeli, čo sa od nich očakáva a ako budú ovplyvnení tým, čo robia iní ľudia. Plánovanie vyžaduje nasledujúce kroky:

Definovanie rozsahu – dekompozícia projektu rozpisom práce (WBS - Work Break-down Structure) identifikáciou fáz a činností projektu. Tím sa môže podieľať na rozložení činností na úlohy.

Definovanie závislostí medzi činnosťami – vyznačenie závislostí v pláne. Vyznačiť treba zvlášť riziká – hlavnú cestu pre sponzorov projektu.

Definovanie činností, ich zoradenie – určenie, ktoré činnosti môžu byť vykonané súčasne.

Odhad úsilia a dĺžky trvania každej činnosti – konzultovať s tímom. Ak to dovoľí časový rozvrh projektu, odporúča sa pridať ku odhadom 15%. Potrebujeme dĺžku trvania v osobohodinách aj celkovú časť úsilia, aby sme ju vedeli vyúčtovať klientom.

Plánovanie zdrojov – stanoviť cenu pre každú činnosť v termínoch osobohodiny, hardvér, softvérové licencie, dodávateľia.

Zostavenie plánu projektu a analýza výsledkov

Získanie súhlasu, pridelených zdrojov a plánu pracovných úloh

Nezávisle od podporných softvérových prostriedkov, použitých na vytvorenie plánu by mal výstupný dokument obsahovať:

- pre každú činnosť meno, ktoré ju vhodne opisuje
- definíciu vstupov činností
- zdroje, ktoré sú pre činnosť potrebné
- dátum začatia a ukončenia činnosti
- súhrnnú cenu v jednotkách ako sú osobohodiny, dodávateľia a fixné ceny ako náklady na hardvér a softvér

Základom plánovania projektov je umenie vedieť čo vypustiť. Je dôležité zamerať sa na kritickú cestu t.j. činnosti, ktoré musia byť dokončené včas, inak sa celý projekt oneskorí. Dôležité sú procesy, ktoré sú jedinečné pre daný projekt, rutinné činnosti môžeme spojiť dokopy. Plán je zároveň prostriedok na odhad nákladov, preto sa používa na komunikáciu. Mal by byť navrhnutý na základe niekoľkých kľúčových princípov:

- Hlavné kontrolné body a vysoko rizikové činnosti sú nápadne vyznačené.

- Postupnosť sleduje reálny čas udalostí čo najpresnejšie.
- Je jasná zodpovednosť za úlohy. Je potrebné vyhnúť sa zoskupeniam nesúvisiacich úloh.
- Plán má byť zrozumiteľný aj ľuďom, ktorí nie sú projektoví manažéri.

Plán je práca, ktorá sa vyvíja, a ktorá bude upravovaná v každom kontrolnom bode. Takto budeme prezentovať plán klientom. Je dôležité pripomenúť im, že zmena je nevyhnutná a zároveň budovať vzájomnú dôveru od začiatočných odhadov, aby projekt skončil úspešne, včas a v rámci rozpočtu.

Ako presne môžu byť pevné štruktúry procesu plánovania, používaného v softvérovom inžinierstve aplikované na stále sa meniace web projekty? Veľa inovátnych projektových manažérov sa naučilo rozbiť výstupy z ich projektov do série „splátok“. Splátky nemusia nutne korešpondovať s klasickými fázami projektu ako návrh, implementácia, testovanie a uvedenie do používania. Namiesto toho zahŕňajú ľubovoľnú časť z požadovanej výstupnej funkcionality, s ktorej odhadom je tím spokojný. Na konci každej splátky klient prijme hodnotnú časť, projektový tím dostane zaplatené a navrhne sa plán na realizáciu ďalšej časti. Web projekt sa stáva komplexným, pretože výsledná cena je pohybujúcim sa cieľom a závisí od miery zmien, ktoré inicializuje klient.

Takýmto „plánovaním za behu“ má klient k dispozícii hrubé odhady ceny a dĺžky trvania celého projektu. A zároveň klient požaduje od projektového tímu iba splnenie termínov a rozpočtu aktuálnej splátky. Tím nie je povinný plniť budúce splátky, pokiaľ nedokončí aktuálnu časť. Môže sa zdať, že tím ľahko vyklúčkuje z projektu. V skutočnosti takýto postup znižuje riziko klienta, pretože klient môže zrušiť projekt na konci ktorejkoľvek splátky. Za predpokladu, že projekt používa štandardné web technológie, klient môže nájsť iný tím, aby projekt dokončil.

Medzi efektívnosťou plánu a počtom úloh, ktoré sme schopní identifikovať je vzťah slabý. Plán je nástrojom komunikácie, ktorý musí vyjadrovať kľúčový význam, ako sú vzájomné závislosti činností a ich nadväznosti. Dobré plány sú vytvorené s pohyblivými časťami, ktoré zodpovedajú zmenám a môžu byť posunuté do nových zostáv bez úplného kolapsu projektu. Web projekty vyžadujú kreatívne riešenia, ktoré prekračujú tradičný prístup centralizovaného plánu. Ten musí byť pevne stanovený predtým, než môžu začať práce na projekte. Vyžadujú rozdelenie fázy plánovania a umožňujú rozvoj dokumentácie prirodzene so zvyškom projektu. S takouto štruktúrou v etapách sa požiadavky zákazníka stanú skôr sústavným dialógom, než položkou na kontrolnom zozname.

6.3 Roly v tímoch

Pri riadení web projektov potrebujeme viesť tím ľudí, ktorí sú experti v rôznych oblastiach. Musíme poznať funkciu každého z nich, ich prínos, správanie a odlišnosti a musíme byť schopní ich motivovať. Zloženie tímu je väčšinou podmienené dostupnosťou pracovníkov, málokedy, ak vôbec, si môže projektový manažér svoj tím vybrať sám. V rôznych organizáciách, ako aj v jednotlivých projektoch, býva rôzne zloženie tímov. Zameriame sa hlavne na web aplikácie, ktorých cieľom je poskytnúť informácie. Tento typ web aplikácií v súčasnosti prevláda. Základné roly v tíme, ktorý vytvára web aplikáciu s cieľom poskytnúť informácie, sú nasledovné:

- Zákazník
- Projektový manažér
- Producent
- Redaktor
- Informačný architekt
- Grafický dizajnér
- Tvorca stránok
- Vedúci programátor
- Programátor
- Databázový expert
- Manažér kvality

Činnosti vykonávané pri tvorbe web aplikácie môžeme rozdeliť do troch skupín: správa obsahu, grafický dizajn a programovanie. Manažér projektu vedie ľudí zo všetkých skupín. Zákazník obyčajne neprichádza do každodenného kontaktu s tímom, on však iniciuje projekt, financuje ho a niekedy aj zaobstaráva potrebné zdroje. Projektový manažér sprostredkováva komunikáciu medzi zákazníkom a vývojovým tímom. Jedna z jeho hlavných úloh je informovať zákazníka o priebehu a aktuálnom stave projektu.

Roly spojené s programovaním zahŕňajú vedúceho programátora, programátora a tvorcu stránok. Roly týkajúce sa obsahovej stránky projektu sú producent a redaktor. Oddelenie grafického dizajnu zahŕňa tvorivého režiséra a grafických

dizajnérov. Informačný architekt sa zaoberá grafickým dizajnom ako aj technológiami. Úzko spolupracuje s programátormi na architektúre aplikácie. Spolu s dizajnérsym tímom zabezpečuje, že rozhrania spĺňajú podmienky použiteľnosti. Manažér kvality je zodpovedný za testovanie všetkých komponentov aplikácie z používateľského hľadiska. Hľadá chyby vo funkcionalite a aj v zobrazovaní obsahu stránok.

6.3.1 Zákazník

Zákazník je ten, kto zadáva zákazku. Môže to byť externý zákazník alebo niekto z vedenia firmy. Zákazník zabezpečuje:

- Koncept projektu
- Rozpočet
- Marketingový plán
- Výtvar a šablónu stránok

Niektoré úlohy zákazník pripravuje spoločne s projektovým manažérom a to najmä v prípade, že sa projekt vyvíja pre vnútornú potrebu firmy. Po začatí projektu nie je zákazník zvyčajne členom vývojového tímu, ale projektový manažér ho pravidelne informuje o stave projektu.

V počiatočných fázach projektu sa treba dohodnúť so zákazníkom na spolupráci, zodpovednosti a určiť spôsob komunikácie a riešenia problémov. Zvyčajne má manažér projektu na starosti tím a aplikáciu, zákazník zabezpečuje obchodné ciele. Tieto činnosti sa niekedy prekrývajú – vtedy presne zadané role dokážu predísť mnohým problémom a urýchľujú projekt. Projektový manažér spolu so zákazníkom odhadujú a sledujú náklady na zdroje. Čím viac znalostí má projektový manažér o obchodných cieľoch zákazníka, tým presnejšie vie tieto náklady odhadnúť a vie tiež pomôcť zákazníkovi v stanovení obchodných cieľov projektu.

6.3.2 Projektový manažér

Manažér projektu zodpovedá za riadenie projektu a za jeho úspešné ukončenie. Spolupracuje so zákazníkom a producentom pri vytváraní požiadaviek na aplikáciu a pri tvorbe rozvrhu a rozsahu projektu. Má základné vedomosti o procese tvorby web aplikácií, avšak jeho špecializáciu a hlavnú náplň práce tvoria rôzne činnosti manažmentu (manažment nákladov, rizík, ľudských zdrojov, komunikácie atď.). Jeho hlavné úlohy sú :

- Tvorba rozvrhu

- Tvorba rozpočtu
- Riadenie projektu s ohľadom na rozvrh, rozsah a požiadavky
- Zabezpečenie zdrojov (priestory, softvér, hardvér atď.)
- Zabezpečenie komunikácie a spolupráce v tíme
- Komunikácia so zákazníkom
- Motivovanie tímu aj jednotlivých členov

6.3.3 Producent

Producent má veľa úloh a množstvo zodpovedností v priebehu projektu. V rôznych firmách býva jeho rola pomenovaná rôzne. Jeho typické úlohy sú:

- Tvorba konceptu projektu
- Zadefinovanie vízie
- Vytvorenie mapy stránok
- Zodpovednosť za výslednú špecifikáciu
- Tvorba rozvrhu
- Tvorba rozpočtu
- Usmerňovanie návrhu
- Usmerňovanie redaktorskej činnosti
- Správa zdrojov pre redaktorov

Producent má zvyčajne bližšie k obsahu a forme prezentácie aplikácie ako projektový manažér. Pri práci na projekte si zachováva pohľad používateľa alebo klienta pri práci na projekte. Vzťah producenta a projektového manažéra je kvôli častému prekrývaniu úloh konfliktný. Je dôležité určiť, kto je za ktoré úlohy a zdroje zodpovedný a oznámiť toto rozdelenie všetkým členom tímu.

6.3.4 Redaktor

Redaktor je zodpovedný za písanie článkov, opisov produktov, nadpisov a iných typov textu. Jeho práca úzko súvisí s prácou producenta. Pri väčších web aplikáciách používa špeciálny nástroj na správu obsahu stránok, textov a obrázkov. Medzi jeho povinnosti patrí:

- Navrhovanie obsahu článkov
- Písanie a dodávanie článkov
- Opis a recenzie produktov
- Zhotovenie záznamov z rozhovorov a stretnutí

6.3.5 Informačný architekt

Informačný architekt zodpovedá za použiteľnosť web aplikácie. Je dobre oboznámený s dizajnom a technológiami. Má zmapované správanie sa ľudí pri rôznych programoch a grafických rozhraniach. Jeho hlavnou úlohou je organizovanie informácií do prirodzenej formy z hľadiska vizuálneho a navigačného.

6.3.6 Grafický dizajnér

Grafický dizajnér vytvára koncept, prejav a atmosféru web aplikácie použitím vhodných obrázkov, ikon, farieb, animácií a typov písma. Má zvyčajne dobré znalosti o používateľskom prístupe k web aplikácii a pri návrhu kladie dôraz hlavne na používateľa. Dizajnéri často úzko spolupracujú s informačným architektom pri návrhu rozmiestnenia stránok, grafického rozhrania a navigačného modelu.

Vo všeobecnosti sú grafickí dizajnéri kreatívni a talentovaní ľudia. Tvorba dizajnu stránok je proces, v ktorom komunikácia zohráva podstatnú úlohu. Dizajnéri spolupracujú so zákazníkom, aby zjednotili svoj a zákaznícky pohľad na dizajn.

6.3.7 Tvorca stránok

Tvorba stránok je založená na jazyku HTML. Tvorca stránok je expert na tento jazyk a iné technológie, ktoré slúžia pre zobrazovanie informácií (napr. JSP Custom Tag Library). Je tiež odborník na štýly, kompatibilitu prehliadačov, rámce (frames) a pod.

Tvorca stránok zhmotňuje návrh dizajnéra do podoby vhodnej na zobrazovanie v prehliadači. Spolupracuje, často paralelne, s grafickým dizajnérom a programátormi. Tvorca stránok musí dobre poznať používané technológie a mať zmysel pre grafický dizajn. Jeho hlavné povinnosti sú:

- Tvorba stránok
- Tvorba šablón
- Vytvorenie modelu aplikácie pre účely návrhu
- Implementácia štýlov

6.3.8 Hlavný programátor

Hlavný programátor sprostredkováva komunikáciu medzi programátormi a projektovým manažérom. Je zodpovedný za vykonávanie prehliadky kódu a usmerňovanie programátorov. Je nesmierne dôležitý hlavne v prípade, že je v projekte prítomný menej skúsený programátor. Jeho hlavné povinnosti sú:

- Vytvorenie technickej špecifikácie
- Prehliadanie kódu
- Riadenie programátorov
- Programovanie

6.3.9 Programátor

Programátor je zodpovedný za vytvorenie biznis logiky aplikácie. Je expert na používané technológie a pri svojej práci spolupracuje s databázovým expertom a s tvorcami stránok.

6.3.10 Databázový expert

Databázový expert je sa zaoberá aspektami projektu, ktoré sa týkajú databáz. Poskytuje svoje vedomosti a skúsenosti programátorom. Na projekte sa podieľa hlavne v jeho počiatočných fázach. Programátori zvyčajne vytvárajú schému databázy na začiatku technického návrhu a túto schému potom konzultujú s databázovým expertom. Databázový expert píše kód špecifický pre použitú databázu ako sú napríklad uložené procedúry (tzv. stored procedures). Väčšina jeho práce spočíva v údržbe databázy a optimalizácii jej výkonnosti.

6.3.11 Manažér kvality

Manažér kvality zodpovedá za testovanie aplikácie. Aplikáciu môžeme testovať na dvoch úrovniach: z hľadiska používateľského rozhrania alebo z programátorského hľadiska. Manažér kvality dobre pozná používané technológie a je schopný nájsť chybu dôkladne zdokumentovať. Ale neopravuje ju. Medzi jeho hlavné povinnosti patrí:

- Informovanie o chybách
- Vytvorenie a správa metodiky manažmentu kvality
- Vytvorenie alebo zaobstaranie prostriedku na informovanie o chybách

6.4 Manažment rozsahu

Štandardné modely vývoja softvéru predpokladajú, že rozsah projektu je presne určený ešte predtým, ako sa projekt začne realizovať. Tento predpoklad však neplatí v prípade vývoja web aplikácií, ktoré sa neustále vyvíjajú. Ich vývoj je podmienený jednak zmenami v technológiách, ale hlavne zmenami v obchodnom prostredí.

Štandardné modely vývoja softvéru sú založené na požiadavkách. Implicitne predpokladajú, že výsledný produkt vývoja bude konfrontovaný s vopred zadefinovanými požiadavkami. Pre tieto modely existujú procesy pre vytvorenie rozvrhu a rozpočtu a analýzu zdrojov, rizík a rozsahu. Aj napriek rozdielom medzi vývojom štandardného softvéru a vývojom web aplikácie je väčšina používaných metód vývoja web aplikácií odvodená zo štandardných metód.

6.4.1 Základný trojuholník - Čas, Náklady, Požiadavky na aplikáciu

Základným predpokladom pre efektívne riadenie rozsahu projektu je skutočnosť, že zákazník musí pochopiť nutné kompromisy medzi nákladmi projektu, časom potrebným na jeho realizáciu a požiadavkami na aplikáciu. Veľa času sa stráca, pokiaľ zákazník nerozumie potrebným kompromisom v rámci tohto trojuholníka. Zákazník sa musí vyjadriť, ktorú z troch osí je ochotný meniť, musí chápať ich vzájomné prepojenie a dôsledok takéhoto rozhodnutia. Úlohou projektového manažéra je tieto vzťahy zákazníkovi vysvetliť a pripomínať a teda stať sa jeho konzultantom pri rozhodovaní.

6.4.2 Požiadavky na zmenu

Aby boli zmeny rozsahu zvládnuteľné, je potrebné, aby všetky požiadavky na zmenu rozsahu boli v písomnej podobe. Súčasťou požiadavky na zmenu rozsahu by mala byť aj relatívna priorita zmeny a dôsledok zmeny na rozvrh a rozpočet. Tento protokol pomáha zákazníčkovi spresniť svoje požiadavky a uvedomiť si ich dôsledky, ale hlavne poskytuje informácie potrebné pri rozhodovaní o riadení projektu.

6.4.3 Správa dokumentov

Správa dokumentov je nevyhnutná súčasť manažmentu rozsahu. Ak členovia tímu nemajú jednoduchý prístup k najnovšej projektovej dokumentácii, nepoznajú skutočný rozsah projektu. Web stránka projektu by mala obsahovať odkazy na najnovšie verzie dokumentov a dátumy posledných zmien. Základné pravidlá pre správu dokumentov sú nasledovné:

- Používanie jednotného pomenovania súborov, ktoré obsahuje autora, verziu a dátum.
- Iba jeden človek je zodpovedný za uchovávanie najnovších dokumentov a udržiavanie aktuálnosti stránky.
- Web stránka by mala obsahovať aj odkazy na predchádzajúce verzie dokumentov.

6.4.4 Iteratívne prístupy

Počas životného cyklu projektu zákazníci získajú nové vedomosti o technológiách ako aj svojich vlastných obchodných potrebách. Obchodné prostredie sa tiež môže za niekoľko mesiacov zmeniť. Kvôli častým zmenám sa veľká pozornosť venuje iteratívnym prístupom k vývoju softvéru, akým je napríklad rýchly vývoj softvéru (Rapid Application Development, RAD).

- Rýchly vývoj softvéru

Metodológia rýchleho vývoja softvéru sa zameriava na vývoj softvéru s dôrazom na rýchlosť vývoja. Je vhodná pre projekty, ktoré majú dobre zadaný rozsah a viditeľný výstup. V ideálnom prípade predpokladá malý tím v úzkom kontakte so zákazníčkovi, ktorý rozhoduje o zmenách vo funkcionalite aplikácie.

Základom RAD je prototypovanie. Vývojári sa snažia vytvoriť čiastočne fungujúci prototyp tak skoro, ako je to možné. Prototyp sa na základe spätnej väzby od

zákazníka v jednotlivých krokoch upravuje. Projektový manažér používajú techniku časových okien (tzv. timeboxing), v ktorej je možné meniť rozsah iterácie, ale dátum odovzdania ostáva nezmenený pre každú iteráciu. Najväčšia výhoda RAD spočíva v tom, že zákazník vidí aplikáciu skoro a môže meniť požiadavky podľa potreby.

- Extrémne programovanie

Jedna z najznámejších aplikácií RAD metodológie je extrémne programovanie. Extrémne programovanie sa sústreďuje na zákazníka, pričom vývojári sa snažia vyvinúť minimum softvéru tak, aby softvér splnil požiadavky zákazníka. Aplikácia sa vyvíja ako postupnosť prototypov, ku ktorým zákazník poskytuje spätnú väzbu. Extrémne programovanie nerozlišuje medzi prototypom a výsledným produktom. Používa tiež niekoľko nových techník, ako napríklad programovanie vo dvojiciach (tzv. pair programming), pre podporu tímovej práce, kreativity a spolupráce medzi vývojármi.

6.4.5 Zhrnutie

Web aplikácie sú odlišné od ostatných kategórií počítačového softvéru. Web inžinierstvo používa iteratívny a inkrementálny proces, pretože čas vývoja web aplikácií je krátky. Všeobecné činnosti, využívané pre manažment projektov v softvérovom inžinierstve, platia pre všetky web inžinierske projekty. Pri ich aplikovaní však musíme brať do úvahy špecifiká vlastné web aplikáciám.

Kapitola 7

Technológie a štandardy

Technická stránka web inžinierstva je postavená na rôznych technológiách. Tieto pokrývajú činnosť klientov, aplikácií strednej vrstvy, serverov a aj vývoj aplikácií. Ale technológie nie sú prakticky použiteľné, pokiaľ nie sú vytvorené štandardy. Štandardy definujú pravidlá a protokoly pre použitie technológií tak, aby klienti, aplikácie a servery mohli navzájom komunikovať a spolupracovať.

V tejto kapitole opíšeme technológie a štandardy používané pri vývoji a implementácii web aplikácií. Nejedná sa úplný a presný výpis všetkých technológií a štandardov, ale o prehľad najzaujímajších, keďže táto doména sa prudko vyvíja a úplný výpis by presahoval rozsah tejto publikácie.

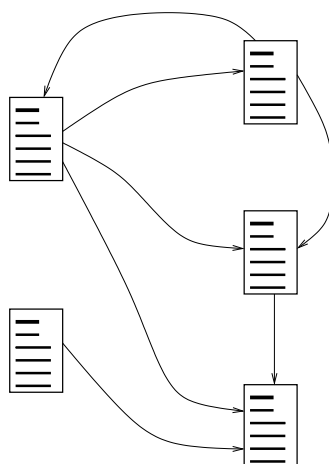
7.1 Klient

Klienti sú pri web aplikáciách predovšetkým prehliadače HTML stránok. Technológie spojené s klientmi sú pravdepodobne najrozšírenejšie a najznámejšie. Ich úlohou je prezentovať používateľovi informácie zo serveru.

7.1.1 Internet

Celosvetová sieť Internet a lokálne siete intranet sú založené na rodine sieťových protokolov TCP/IP, ktoré fungujú na princípe prepájania paketov. Správy medzi jednotlivými účastníkmi siete sú rozdelené na pakety a každý paket sa v sieti prenáša nezávisle od ostatných. Tento princíp spolu s decentralizáciou (neexistuje jediný centrálny bod, ktorý by prepojoval všetkých účastníkov) umožňuje stavať robustné siete.

Protokoly TCP/IP zabezpečujú iba prenos správ. Pre činnosť aplikácií je to však nedostatočné, preto existujú protokoly, ktoré používajú TCP/IP a poskytujú



Obrázok 7.1: Hypertextová štruktúra stránok

aplikáciám dodatočné služby. Sú to napríklad protokoly:

HTTP – prenos súborov a informácií pre systém WWW.

FTP - starší systém prenosu súborov.

SSH - kryptograficky zabezpečený prístup ku terminálovým aplikáciám.

7.1.2 Hypertext

„Hypertextom je označovaný systém pre prezentáciu textových informácií, ktorý umožňuje prepojiť textové stránky, nesequenčným, úplne všeobecným spôsobom.“ [Jel98]

Systém WWW stavia na stránkach prepojených odkazmi (obrázok 7.1), teda na koncepte hypertextu. Tento koncept nie je nový a využíval sa už aj v slovníkoch, encyklopédiách a v odkazoch na literatúru. Jeho výhodou je, že umožňuje vyjadriť väzby v opisovaných myšlienkach, ktoré sa dajú iba ohraničene alebo obtiažne vyjadriť prostredníctvom lineárneho textu.

Podrobnejšie sa tvorbe a využitiu hypertextových dokumentov venujú kapitoly 1.6, 3.1 a 4.2.

7.1.3 Značkovacie jazyky

Značkovacie jazyky vznikli z potreby konzistentne vyznačovať úseky textu a dopĺňať do neho prídavné informácie. Prídavné informácie sa v nich reprezentujú

pomocou značiek.

Jedným z prvých značkovacích jazykov bol jazyk GML, ktorý sa používal vo firme IBM pre uchovávanie právnických textov. Zovšeobecnením a zdokonalením tohto jazyka vznikol nový jazyk SGML. Jazyk SGML [ISO86] je veľmi všeobecný, umožňuje napríklad vlastnú definíciu sady značiek pomocou definícií typov dokumentov DTD¹ a má veľa voliteľných parametrov. S tým spojená komplexnosť však spomalila jeho rozšírenie.

Jednou z aplikácií SGML je aj jazyk HTML [W3Cb], ktorý sa používa na opis WWW stránok. Sady značiek sú pevne definované pre každú verziu HTML jazyka pomocou DTD a opisujú elementy stránky prevažne z vizuálneho hľadiska. Postupom času sa však ukázalo, že žiadna verzia HTML jazyka svojou sadou značiek nestačí na všetky účely a bolo by najlepšie, keby bolo možné definovať vlastné sady značiek, tak ako v jazyku SGML.

Riešenie tohto problému poskytol jazyk XML [W3Ca] [Kos00], ktorý poskytuje používateľovi možnosť definovať si vlastné značky pomocou DTD tak ako SGML, ale je podstatne jednoduchší ako SGML. Prísnejšia syntax zápisu tak isto umožňuje jednoduchšie vytvárať aplikácie ktoré generujú alebo spracúvajú informácie zapísané v XML.

7.1.4 HTML

Jazyk HTML umožňuje zapisovať formátovacie a štruktúrovacie informácie do textového dokumentu. Taktiež pomocou notácie URI [BFIM98] podporuje tvorbu hypertextových odkazov v rámci dokumentu, odkazov na iné HTML dokumenty, odkazov na rôzne multimediálne informácie a všeobecné binárne súbory. Sústava takto prepojených dokumentov v sieti Internet tvorí informačný systém World Wide Web.

Značky a elementy

Elementy sú štruktúry, ktoré opisujú časti HTML dokumentu. Napríklad element `p` reprezentuje odstavce, element `em` definuje zvýraznenie časti textu. Element má tri časti: počiatočnú značku, obsah a ukončovaciu značku. Značka je osobitný text, „vyznačenie“, ktoré je ohraničené znakmi `<`, `>`. Napríklad element `em` má počiatočnú značku ``, a ukončovaciu značku ``. Počiatočná a ukončovacia značka ohraničujú *obsah* elementu `em`.

```
<em>Tu sa nachádzadza zvýraznený text.</em>
```

¹DTD je notácia pre zápis gramatík.

V jazyku HTML nie sú mená elementov závislé od veľkosti znakov, teda ``, ``, `` sú rovnaké mená. Elementy sa nesmú navzájom prekrývať, ale môžu sa navzájom vnárať. Pokiaľ sa počiatočná značka elementu `em` vyskytne v elemente `p`, musí sa ukončovacia značka elementu `em` nachádzať tiež v elemente `p`.

```
<em> ... <p> nepovolené prekrývanie </em> ... </p>
```

```
<em> ... <p> povolené vnáranie </p> ... </em>
```

Niektoré elementy dovoľujú vynechať počiatočnú alebo ukončovaciu značku. Napríklad ukončovacia značka elementu `li` je vždy nepovinná, lebo koniec elementu `li` je indikovaný začiatkom nasledujúceho elementu `li` alebo koncom zoznamu (značkou ``)

```
<ul>
  <li>Prvá položka zoznamu, žiadna ukončovacia značka
  <li>Druhá položka zoznamu, obsahuje nepovinnú
  ukončovaciu značku</li>
  <li>Tretia položka zoznamu, žiadna ukončovacia
  značka
</ul>
```

Niektoré elementy nemajú ukončovaciu značku, lebo nemajú žiaden obsah. Tieto elementy, napr. element `br` pre zalomenie riadku, sú reprezentované iba počiatočnou značkou a sú označované ako prázdne.

Atribúty

Atribúty elementu definujú rôzne vlastnosti elementu. Napríklad element `img`, ktorý definuje obrázok na stránke, potrebuje URL umiestnenie obrázku zapísané v atribúte `src` a alternatívny text zapísaný v atribúte `alt`.

```

```

Atribúty sa nachádzajú iba v počiatočných značkách elementov, nikdy v ukončovacích značkách. Majú tvar *meno-atribútu*="hodnota-atribútu". Meno atribútu nie je závislé od veľkosti znakov, ale hodnota môže byť závislá. Hodnota atribútu je ohraničená jednoduchými alebo dvojitémi úvodzovkami. Tieto úvodzovky nie sú povinné, pokiaľ sa hodnota skladá iba z písmen a-z a A-Z, číslic 0-9, znaku mínus a bodky.

Špeciálne znaky

Niektoré znaky sú v jazyku HTML rezervované pre vyznačovanie, nesmú sa vyskytovať priamo v texte. Znak `<` môže byť reprezentovaný entitou `<`; Podobne znak `>` môže byť zapísaný `>`; a znak `&` je zapísaný ako `&`;

Alternatívou k entitám je použitie číselných odkazov na znaky. Ľubovoľný znak môže byť zapísaný pomocou jeho číselného odkazu na základe jeho kódu v znakovej sade ISO 10646. Napríklad sa dá použiť `©` pre znak © alebo `λ` pre znak λ.

Komentáre

Komentáre v jazyku HTML majú komplikovanú syntax, ktorá sa ale dá zjednodušiť pravidlom: komentár treba začať pomocou `<!--`, zakončiť pomocou `-->` a nepoužívať v komentári `--`.

```
<!-- Toto je poznámka -->
```

Úplný HTML dokument

Dokument začína tzv. DOCTYPE deklaráciou. Táto deklaruje verziu jazyka HTML, ktorú dokument spĺňa. Nasleduje HTML element, ktorý obsahuje elementy HEAD a BODY. HEAD obsahuje informácie o dokumente, ako napríklad titul alebo kľúčové slová. BODY obsahuje samotný obsah dokumentu, zložený z elementov a textu.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN"
  "http://www.w3.org/TR/REC-html40/strict.dtd">
<html>
  <head>
    <title>Titul dokumentu</title>
  </head>
  <body>
    <h1>Hlavný nadpis</h1>
    <p>odstavec <em>zvýraznené</em></p>
    <p>d ďalší odstavec</p>
    <ul>
      <li>položka zoznamu</li>
      <li>d ďalšia položka zoznamu</li>
    </ul>
  </body>
</html>
```

7.1.5 Multimédia

Multimédia sú rôzne digitalizované informácie, určené pre vnímanie človekom. Systém WWW podporuje multimédia vo forme obrazových, zvukových a video informácií, pričom v každej oblasti existujú viaceré technológie a štandardy.

Obraz

Historicky najstarší formát používaný vo WWW pre uchovávanie dvojrozmerných bitmapových obrazov je grafický formát GIF, ktorý na kompresiu údajov používa kódovanie LZW. Novší grafický formát PNG [Bou97] dosahuje lepšie výsledky ako GIF (vyššia kompresia) a rieši niektoré jeho nedostatky (podpora čiastočnej priehľadnosti, nie je obmedzený na 256 farieb).

Oba uvedené formáty komprimujú obraz bezstratovo (po dekompresii sa získa pôvodný nezmenený obraz), ale sú efektívne iba pre obrazy s malým počtom farebných prechodov. Pre plne farebné obrazy s veľkým počtom farebných prechodov boli vyvinuté formáty a komprimačné techniky, ktoré využívajú nedokonalosti v ľudskom vnímaní obrazu a nepotrebné informácie odstraňujú. Výsledkom je stratová kompresia, ktorá nereprodukuje obraz absolútne presne, ale s malými a vo väčšine prípadov tolerovateľnými odchýlkami. Najznámejší formát tohto druhu je JPEG [ISO94], ktorý používa kompresnú techniku DCT. Technika DCT umožňuje zvoliť kompromis medzi veľkosťou komprimovaného obrazu a kvalitou jeho reprodukcie. Nástupcom formátu JPG je formát JPEG 2000, ktorý používa vlnkové (tzv. wavelet) transformácie.

Obraz je možné opísať nielen ako bitovú mapu, ale aj pomocou geometrických útvarov. Tento opis je väčšinou stručnejší ako bitová mapa a obraz je možné kvalitne reprodukovať s ľubovoľným zväčšením. Geometrický opis používajú vektorové grafické formáty SVG [W3C01] a PDF.

Zvuk

Najrozšírenejší formát pre ukladanie zvukových záznamov je MP3 (presnejšie MPEG-1 layer 3), ktorý používa stratovú kompresiu. Ďalší obľúbený formát je OGG. Spolu s kompresiou Vorbis dosahuje lepšie výsledky ako MP3. Oba formáty umožňujú zvoliť kompromis medzi veľkosťou komprimovaného záznamu a kvalitou jeho reprodukcie.

Formát MIDI používa pre záznam techniku podobnú notovému zápisu, pomocou ktorej je možné skladby ukladať oveľa efektívnejšie ako v prípade MP3 alebo OGG. Na druhej strane neumožňuje uložiť ľubovoľný zvukový záznam.

Video

Formáty MPEG-1 [ISO93], MPEG-2 používajú algoritmus DCT rozšírený o predspracovanie, ktoré využíva podobnosť snímok video záznamu. MPEG-1 sa používa pre uchovávanie obrazu a sprievodného zvuku na pamäťové médium, MPEG-2 pre jeho efektívny prenos.

Najnovší z MPEG formátov je MPEG-4 [ISO99]. Zaoberá sa efektívnym kódovaním a kompresiou širokej palety multimediálnych prvkov. Pokrýva statický obraz, pohyblivý obraz, dvojrozmernú vektorovú grafiku, text, trojrozmernú grafiku, všeobecné zvukové záznamy a syntetizovaný zvuk. Na kompresiu statického a pohyblivého obrazu sa používajú aj DCT aj vlnkové transformácie, pričom v prípade pohyblivého obrazu sa pre lepšiu kompresiu ešte využíva podobnosť po sebe idúcich snímok. Syntetizovaný zvuk vychádza z formátu MIDI, ale poskytuje oveľa väčšie možnosti, ako napríklad syntézu reči. MPEG-4 umožňuje pomocou jazyka BIFS, ktorý vychádza z VRML, kombinovať všetky vyššie uvedené prvky do jednotného celku.

Na záznam pohyblivého obrazu (prevažne pod platformou Windows) sa zaužíval formát AVI, v ktorom sa pomocou vymeniteľných modulov dajú používať rôzne kompresné techniky. V súčasnosti najobľúbenejšie moduly (napr. DivX, FFmpeg, XviD) používajú kompresné algoritmy z formátu MPEG-4.

Trojrozmerná grafika

VRML97 [ISO97] je formát pre plošný opis trojrozmerných objektov a scén. Bol vytvorený aj pre tvorbu systémov virtuálnej reality, preto okrem opisu tvaru a povrchu objektov obsahuje prostriedky pre animáciu, interakciu a manipuláciu s objektami a umožňuje spoluprácu so skriptovacími jazykmi. Podobne ako formát HTML umožňuje vytvárať spojenia medzi stránkami, VRML umožňuje tvoriť spojenia medzi virtuálnymi svetmi.

7.2 Server

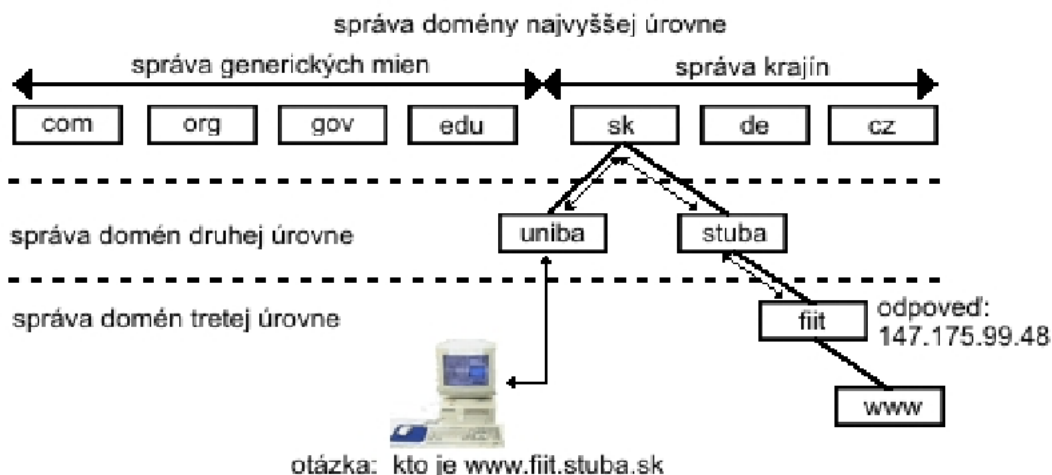
Server tvorí významnú súčasť dnešných dynamických klient-server riešení. Pojem môže byť zavádzajúci, pretože ním označujeme hardvér, tak ako aj aplikáciu. V našom kontexte budeme hovoriť o aplikáciách, ktoré spracúvajú dopyty klienta. Aplikácií na strane servera je veľké množstvo, preto sme sa rozhodli prezentovať tie najpoužívanejšie.

7.2.1 DNS (Domain Name System)

Každý projekt na Internete by mal byť prístupný prostredníctvom svojho ľahko zapamätateľného mena. Systém doménových mien využíva prevod IP adresy na lepšie zapamätateľné meno. Tento spôsob umožňuje prístupit' na rovnaký počítač dvoma spôsobmi.

- prostredníctvom IP adresy (napr. 147.175.111.112, 2477748080)
- prostredníctvom doménového mena (napr. `fornax.elf.stuba.sk`)

Doménové mená sú pridelované hierarchicky. Celú doménu spravuje server najvyššej úrovne. V našom prípade je to doména `sk`. Ten sa odkazuje na server prvej úrovne (z toho názov, doména prvej úrovne). Server prvej úrovne sa odkazuje na server druhej úrovne atď. Uvedený spôsob je efektívny pre spracovanie doménových mien (RFC 1035), ale aj na prevod IP adresy z doménového mena.



Obrázok 7.2: Prevod doménového mena na IP adresu

Problematická je voľba doménového mena (RFC 1034) pri uvedení projektu na Internet. Doménové meno vychádza z mena projektu alebo spoločnosti, ktorá ho nasadzuje. Koncovku volíme podľa zamerania aplikácie generickú, alebo priradenú priamo krajine. Obľúbenú generickú koncovku com (commercial) môže použiť ľubovoľný záujemca na svete. Naopak koncovka priradená priamo krajine býva často pridelená iba jej obyvateľovi alebo spoločnosti danej krajiny. Meno sa musí ľahko pamätať a je vhodné, aby bolo pre potencionálnych používateľov intuitívne. Pri príliš dlhom mene je potrebné zaviesť aj jeho skrátenú podobu a naučiť ju používať pravidelných používateľov, napríklad automatickým presmerovaním na skrátené meno. Ako pekný príklad uvádzame slovenský vyhľadávač SuperZoznam (www.superzoznam.sk) a jeho skrátenú doménu (www.sz.m.sk).

7.2.2 Elektronická pošta

Súčasný informačný systém určený do prostredia Internetu často využíva k automatickej komunikácii s používateľom služby elektronickej pošty. Bežný používateľ sa stretáva najmä s komunikáciou zo strany servera, napríklad pri zabudnutom hesle, prijímaní elektronickej informácie a pod. Spracovanie opačnej komunikácie nie je až také známe, napriek širokému použitiu najmä v komerčných systémoch. Ide o automatické spracovanie elektronickej pošty a jej archiváciu (napr. konferencia prostredníctvom elektronickej pošty).

K spracovaniu elektronickej pošty využívame dve skupiny protokolov:

- Prijímanie správ: IMAP, POP3
- Odosielanie správ: SMTP

IMAP4

IMAP4 (Internet Message Access Protocol, RFC 1730) je novšia služba ako POP3. Umožňuje triediť správy na strane servera. Hlavnou výhodou je, že na server môžeme pristupovať z viacerých klientov, pričom sa správy štandardne nenahrávajú do klienta. Klient prečíta iba hlavičky a na ich základe si môže používateľ vybrať, ktoré správy bude čítať.

POP3

POP3 (Post Office Protocol 3, RFC 1939) je starší štandard, ktorý sa časom osvedčil pre automatické spracovanie elektronickej pošty. Je doplnený o šifrovaný prístup a bezpečnú autentifikáciu. Server POP3 štandardne čaká na porte 110, prípadne na porte 995 (šifrovaná komunikácia).

Ukážka komunikácie medzi klientom a serverom pri prijímaní pošty prostredníctvom protokolu POP3. Používateľský reťazec je označený U a odpoveď servera je označená S.

```

U: telnet server.sk 110
S: Trying 127.0.0.1...
S: Connected to server.sk.
S: Escape character is '^]'.
S: +OK
U: user singer
S: +OK
U: pass mojeheslo
S: +OK
U: list
S: +OK
S: 1 22325
S: 2 1583
S: .
U: retr 2
S: +OK
S: Return-Path: <singer@pobox.sk>
S: Received: from seldon.terminus.sk (root@seldon.terminus [10.42.42.130])
S: by fornax.elf.stuba.sk (8.12.9/8.12.4) with ESMTP id i3P8cLwf002034
S: for <singer@fornax.sk>; Sun, 25 Apr 2004 10:38:21 +0200
S: Received: from www4.pobox.sk (www4.pobox.sk [212.5.216.14])
S: by www4.pobox.sk (8.12.10/8.12.10) with SMTP id i3P8bK5o029030
S: for <singer@fornax.sk>; Sun, 25 Apr 2004 10:37:20 +0200
S: Date: Sun, 25 Apr 2004 10:37:20 +0200
S: Message-Id: <200404250837.i3P8bK5o029030@www4.pobox.sk>
S: From: Martin Spevak <singer@pobox.sk>
S: Subject: ukazka pre webing
S: To: singer@fornax.sk
S:
S: sprava je WEB engineering
S: .
U: quit
S: +OK
S: Connection closed by foreign host.

```

SMTP

SMTP (Simple Mail Transfer Protocol, RFC 821) je služba na odosielanie správ elektronickej pošty. Server čaká na porte 25. Jeho úlohou je prevziať správu a odoslať na definované adresy. Vzhľadom na rozdielne protokoly používané pri prenose, odosielaná správa nesmie obsahovať binárne údaje. Preto musia byť všetky prílohy konvertované do čistého ASCII. Na konverziu sa používajú dva prístupy.

Staršia metóda binárnych údajov používa 7 bitové kódovanie. Má širokú podporu v poštových klientoch. Konverziu do uvedeného formátu je možné vykonať príkazom uuencode a spätnú konverziu príkazom uudecode. Veľa používateľov nedokáže identifikovať údaje po prijatí pošty v takomto kódovaní. Stačí však celú oblasť skopírovať do zvláštneho súboru a použiť spomenuté príkazy. Ukážka

kódovaného súboru:

```
begin 644 pipe.txt
M/B! : 86II ; 6%L ; R!B > 2 ! M92P@ : F5S = & QI ( & IE ( & UO > FYE ( ' ) O > G9E = ' 9I = " ! D
M871A ( ' 8 @ < & EP92X @ 3F % P < FEK ; & % D ( & MD > 7H @ " CX @ 8G5D = 2 ! Z < ' ) A8V ] V879A
. . .
; + BXI ( ' P @ < V ] R = " ` N + BX @ / B ` D5EE35 % 50 " @ H *
`
end
```

Novšie metódy sú všetky zahrnuté v takzvanom MIME (Multipurpose Internet Mail Extensions, RFC 2045, RFC 2046, RFC 2047, RFC 2048, RFC 2049). Kódovanie je optimalizované pre prenos textu, aplikácií, správ, obrázkov, audia či videa. Volí sa automaticky pri rozoznaní typu prenášaných údajov.

Základné príkazy pre SMTP:

HELO <doména> Príkaz prihlásenia na poštový server.

MAIL FROM: <zdrojová adresa> Určenie adresy odosielateľa

RCPT TO: <cieľová adresa> Určenie adresy adresáta

DATA Zápis obsahu správy. Môže obsahovať nepovinnú časť, dokončenie hlavičky. Nepovinná časť sa oddeľuje od samotného obsahu správy prázdny riadkom. Správa musí byť ukončená bodkou na samostatnom riadku.

QUIT Ukončenie práce so serverom. Server ukončí TCP spojenie.

Ukážka komunikácie medzi klientom a serverom. Používateľský retázec je označený U a odpoveď servera je označená S.

```
U: telnet fornax.sk 25
S: Connected to fornax.sk.
S: Escape character is '^]'.
S: 220 fornax.elf.stuba.sk ESMTP Sendmail 8.12.9/8.12.4; Sun, 25 Apr 2004 12:53:42
S: +0200
U: helo fornax.sk
S: 250 fornax.elf.stuba.sk Hello singer@fornax.elf.stuba.sk [147.175.111.112], pleased to meet you
U: mail from: singer@fornax.sk
S: 250 2.1.0 singer@fornax.sk... Sender ok
U: rcpt to: singer@pobox.sk
S: 250 2.1.5 singer@pobox.sk... Recipient ok
U: data
S: 354 Enter mail, end with "." on a line by itself
U: Subject: testovací email
U:
U: obsah emailu
U:.
S: 250 2.0.0 i3PArgwf003139 Message accepted for delivery
U: quit
S: 221 2.0.0 fornax.elf.stuba.sk closing connection
S: Connection closed by foreign host.
```

Pri odosielaní správy je do hlavičky pripojený jednoznačný identifikátor. Klienti pre spracovanie elektronickej pošty ho vkladajú do hlavičky pri odpovedi na sprá-

vu ako referenčný identifikátor. Túto vlastnosť využívajú napríklad konferencie prostredníctvom elektronickej pošty pri určovaní stromu odpovedí.

Automatické spracovanie doručenej pošty

Nemenej dôležitým je automatické spracovanie elektronickej pošty. Každý väčší poštový server umožňuje jej spracovanie prostredníctvom externých aplikácií alebo dokonca integrovaným skriptom. Spracovanú poštu môže systém triediť, preposielať, prípadne na ňu automaticky odpovedať.

Jedným riešením automatického spracovania doručenej pošty je aplikácia s názvom procmail[proa]. Je známa najmä používateľom systémov UNIX. Jednoduchým spôsobom dokáže predspracovať správu či už na základe adresy, obsahu, príloh alebo ju odovzdať na spracovanie externej aplikácii .

Konfigurácia spracovania prichádzajúcich správ sa zapisuje do špeciálneho súboru s názvom procmailrc:

```
SHELL=/usr/bin/bash
MAILDIR=/home/singer/mail
LOGFILE=$MAILDIR/procmail.log
#spracovanie externou aplikáciou
:0:
To.*si99@fornax.*sk
—/home/singer/bin/readMail3
#preposlanie na inú adresu
:0:
To.*BUGTRAQ@securityfocus.com
!singer@2icommerce.com
#zapísanie do špeciálneho súboru
:0:
To.*konference@java.cz
java
```

7.2.3 HTTP

Protokol HTTP (Hypertext Transfer Protocol 1.1, RFC 2616, RFC 2068) verzie 1.1 je rozšírením verzie 1.0 (RFC 1945). Server podporujúci HTTP protokol nazývame web server. Požiadavky spracováva na porte 80.

Protokol slúži na prenos údajov, na ktoré sa záujemca odkazuje prostredníctvom URL. Jeho schému môžeme opísať ako:

```
"http:" "://" host' [ ":" port ] [ absolútna_cesta [ "?" parametre ] ]
```

Je dôležité si uvedomiť, že časť po číslo portu nie je citlivá na veľkosť znakov. Keďže sa jedná o doménové meno, ktoré je transformované na IP adresu a toto meno sa bude vždy transformovať na malé písmená. Štandardne sa doménové meno píše malými znakmi. Časť uvedená po porte závisí od veľkosti znakov.

Nová verzia protokolu umožňuje napríklad perzistentné pripojenie, kompresiu prenášaných údajov, ovládanie virtuálnych hostov a veľmi dôležité označovanie kódovej stránky prenášaných údajov.

Základom ostáva prenos údajov spôsobmi GET a POST. Metóda GET prenáša údaje ako súčasť URL. Metóda POST posiela svoje údaje v rámci tela dopytu, navyiac umožňuje prenos binárnych údajov. Po spracovaní musí server vrátiť návratový kód, ktorým je možné identifikovať prípadnú chybu.

Nasleduje ukážka komunikácie medzi klientom a serverom. Používateľský reťazec je označený U a odpoveď servera je označená S.

```
U: telnet fornax.sk 80
S: Trying 147.175.111.112...
S: Connected to fornax.sk.
S: Escape character is '^]'.
U: GET / singer/msi.php?param=get_param HTTP/1.1
U: Host: fornax.sk
U:
S: HTTP/1.1 200 OK
S: Date: Sun, 25 Apr 2004 13:34:45 GMT
S: Server: Apache/1.3.27 (Unix) mod_ssl/2.8.14 OpenSSL/0.9.6e PHP/4.3.3
S: X-Powered-By: PHP/4.3.3
S: Transfer-Encoding: chunked
S: Content-Type: text/html
S:
S: 3e
S: <html>
S: <body>
S: prijaty text: "get_param"
S: </body>
S: </html>
S:
S: 0
```

7.2.4 Web server

Web server komunikuje so svojim okolím prostredníctvom protokolu HTTP. Keďže sme si už základnú funkčnosť vysvetlili v prechádzajúcej časti, môžeme sa bližšie pozrieť na menej známe funkcie servera. Rozšírené vlastnosti sú podpora modulov, práca s tzv. session, správa virtuálnych domén a podpora automatického spracovania údajov externými entitami.

Podpora modulov

Súčasný web server (Apache, IIS atď.) umožňuje rozšíriť svoje vlastnosti prostredníctvom modulov. Moduly môžu byť implementované v ľubovoľnom jazyku, ale musia dodržiavať aplikačné rozhranie pre daný server. Pridávajú do servera nielen nové funkcie, ale často aj celý programovací jazyk.

Server dokáže zabezpečiť zdieľanie údajov medzi jednotlivými modulmi a prí-

padne aj externými aplikáciami (COM objekty v IIS, Apache - posielanie správ, využívanie spoločnej pamäte). Server môže inicializovať modul pri štarte a ten je spustený počas celej jeho prevádzky a globálne premenné môžu byť známe pre všetky inštancie servera. Tento spôsob využitia modulov priniesol nové možnosti ako je napríklad perzistentné spojenie s databázou alebo zdieľanie spoločnej pamäte medzi aplikáciami.

Session

Môžeme povedať, že až definícia session (RFC 2543) umožnila vytváranie plnohodnotných web aplikácií. Primárnou úlohou session je identifikácia klienta jednoduchou a najmä jednoduchou formou. Z toho vyplýva aj možnosť presunúť časť logiky z klienta na server. Výrazne sa zvýšila aj bezpečnosť, pretože údaje, ktoré sa posielali cez formuláre sa museli pri každom kroku kontrolovať a aj malá chyba sa mohla ľahko vypomstíť pri ukladaní údajov do databázy. Programátor určí, ktoré hodnoty sa majú uchovávať v rámci session, pričom po ich inicializácii sa už o ďalšie spracovanie nemusí starať. Súčasťou môže byť udržiavanie spojenia na databázu. Server zabezpečuje, že konkrétnemu klientovi sa priradí práve jeho databázové spojenie, ktoré využíva napríklad svoje dočasne vytvorené tabuľky.

Automatické spracovanie

Je využívané najmä robotmi prechádzajúcimi Internet. Môže ísť o vyhľadávacie služby, alebo robotov plniacich istú špecifickú úlohu. Problém môže nastať, ak nechceme aby nám z nejakého dôvodu robot stránku spracovával. Dôvody môžu byť rôzne, od obsahu interných informácií, až po duplikovanie stránok na viacerých adresách. Kritériá, ktorými sa majú riadiť takéto roboty môžeme zadať do špeciálneho súboru robots.txt, ktorý umiestnime na adresu `http://domena/robots.txt[rob]`.

Jednotlivé záznamy začínajú:

```
User-agent: <meno> ,
```

kde <meno> určuje meno robota, ako sa identifikuje. Hodnota „*” je platná pre všetky roboty. Ďalej môžeme zadať relatívne adresy, ktoré nemajú byť prehliadané:

```
Disallow: /prvy
```

Nevýhodou je, že sa musíme spoliehať na akceptovanie súboru robotom, pretože v opačnom prípade sú údaje samozrejme poskytnuté. Ako sme spomenuli, súbor je uložený iba na jednom mieste, takže je jeho úprava v prípade viacerých

používateľov systému obtiažna. Alternatívou je META tag [spi]

```
<META NAME="ROBOTS" ...>
```

ktorý sa umiestňuje priamo do HTML súboru.

7.2.5 Proxy server

Pôvodnou úlohou proxy serverov bol efektívnejší prenos informácií medzi klientom a web serverom. Mohli by sme ho prirovnať k internetovej vyrovnávacej pamäti (cache). Ak viacero používateľov používa jeden proxy server, určitý dokument sa sťahuje z Internetu iba raz. Následne je prístupný prostredníctvom proxy servera až do uplynutia časového limitu.

Z pohľadu web aplikácií je zaujímavejšou vlastnosťou proxy servera filtrovanie. Proxy server dokáže filtrovať informácie určitého typu (napr. súbory java s príponou class) alebo druhu (stránky obsahujúce určitý text). Pri projektovaní web aplikácie generického charakteru musíme vedieť, či daná spoločnosť z bezpečnostných dôvodov nefiltruje konkrétny typ údajov.

7.2.6 SQL server

SQL (Structured Query Language) server tvorí dôležitú súčasť každého väčšieho projektu. Pomerne efektívne ukladanie údajov a prístup k nim umožnilo značné rozšírenie databáz. Medzi najrozšírenejšie databázy patria relačné (postavené na relačnej algebre, tzv. RDBMS), objektové (rozšírenie relačnej databázy o objekty) a nový typ databáz pre podporu OLAP a dolovania v dátach (MS SQL) s rozšírením o štatistický výber.

K údajom uloženým na serveri pristupujeme cez dopytovací jazyk (SQL). Jeho prvá verzia bola štandardizovaná už v roku 1986 (ANSI SQL86). Druhá, v dnešnej dobe najpodporovanejšia verzia bola prijatá v roku 1992 (ANSI SQL92, alebo SQL2), aj napriek štandardu prijatému v roku 1999 (SQL99).

Základným problémom, najmä pri generickom softvéri, je podpora spolupráce s ľubovoľnou databázou. Môžeme využiť dva prístupy a to dodržiavanie štandardu, alebo využitie aplikačného rozhrania.

SQL92 (ISO/IEC 9075:1992)

Je štandardom, ktorý implementujú takmer všetky databázy. Bol prijatý už v roku 1992 a definuje množinu SQL príkazov, ktoré musia byť implementované. Pri dodržaní predpísaných konštrukcií tohto štandardu pri vývoji web aplikácie, je

pravdepodobný jej bezproblémový chod na väčšine dnešných databázových systémoch.

ODBC (Open Database Connectivity)

ODBC je štandardizované rozhranie pre prácu s ľubovoľným podporovaným prostredím prostredníctvom SQL. Prostredím môže byť štandardná databáza, ale aj súborový systém či tabuľkový procesor. Prostredie s podporou rozhrania ODBC dokáže komunikovať prostredníctvom tohto protokolu na niekoľkých úrovniach. Podporovaný protokol je presne špecifikovaný v ODBC SDK. Základná úroveň je spoločná pre všetky databázy. Najvyššia úroveň má podporu na konkrétne typy databáz alebo iba na jedinú. Pri využití ODBC aplikácia nepozná s akou databázou pracuje. V prípade, že daná databáza nepodporuje konkrétne funkcie, emulujú sa v ovládači ODBC. Tento spôsob umožňuje využívať napríklad transakcie v databáze DBF, rovnako ako v databáze PostgreSQL.

Problémom ODBC je, že nezaručuje štandardne očakávané spracovanie SQL príkazu ani pre prvú úroveň. Príkladom je údajový typ serial v kombinácii s databázou DBF, ktorá ho štandardne nepodporuje. Údajový typ nie je emulovaný ovládačom ODBC, preto môže prísť k neočakávaným situáciám.

JDBC (Java Database Connectivity)

JDBC je podpora univerzálnej spolupráce s databázami v programovacom jazyku Java. Rovnako ako ODBC umožňuje písanie aplikácií tak, aby pri dodržaní pravidiel a pri následnej zmene databázy stačilo zmeniť príslušný ovládač, prostredníctvom ktorého sa aplikácia dopytuje na SQL server. Podporuje tri úrovne pre spracovanie dopytu klienta, pričom každá úroveň býva umiestnená v samostatnom balíčku.

7.2.7 Bezpečnosť na strane servera

Bezpečnostná stratégia tvorí neoddeliteľnú súčasť každého projektu a zvlášť projektu umiestneného v prostredí Internetu. Na strane servera existuje niekoľko možností, ako znížiť bezpečnostné riziko pri prenose a spracovaní údajov.

Web server

Pri prenose citlivých údajov môžeme využiť protokol HTTPS, ktorý je doplnením protokolu HTTP o šifrovanie. Používa sa 128 bitový kľúč, čo je postačujúce pre väčšinu údajov. Server spracováva šifrované požiadavky na inom porte ako klasické HTTP. Pretože sa všetky údaje prenášajú v čistom texte, musíme na protokol HTTPS

prejsť ešte pred prenosom citlivých údajov. Nevýhodou je pomalší prenos údajov, preto je vhodné po skončení prenosu citlivých dát prejsť na klasický protokol.

Bezpečnosť sa výrazne zvýši pri správnom použití mechanizmu session. Presunutím logiky aplikácie z klienta na server zabránime opätovnému prenosu rovnakých údajov.

Proxy server

Je dôležité uvedomiť si, že proxy server ukladá celé stránky aj s prípadnými informáciami, ktoré nemajú byť zverejnené. Tento spôsob umožňuje odchytiť napríklad identifikátor session, ktorý sa vo väčšine systémov už inak nekontroluje. Na strane web servera musíme zabezpečiť, aby sa takého stránky neukladali. Docielime to napríklad zápisom informačných údajov do hlavičky. Nastavíme platnosť stránky na aktuálny čas, čím sa stane pre proxy server neplatná a nebude ju ukladať.

SQL server

Údajový server tvorí najpoužívanejšiu súčasť systémov pracujúcim v prostredí Internetu. Okrem toho, že niektoré databázy podporujú šifrovanie údajov, musíme určité časti šifrovať samostatne. Väčšina používateľov pristupuje k rôznym systémom s rovnakými heslom. Dôležité je šifrovanie hesla pomocou jednosmerného algoritmu (napríklad MD5), kde overenie prebieha v zašifrovaní prijatého hesla a porovnaní šifrovaných reťazcov. V prípade, že používateľ heslo zabudne, systém môže odoslať prvé a posledné písmeno hesla ako pomôcku.

Záver

Deväťdesiate roky boli doménou osobných počítačov, ktoré vzájomne nespolu-pracovali. Boli to samostatné jednotky, ktoré mali spočiatku dokonca len textové rozhranie. Tam niekde sú počiatky dnešného softvérového priemyslu, na rozdiel od veľmi dobre rozvinutého priemyslu hardvérového. Spoločnosti vyrábajúce hardvér mali rozdelený softvérvý priemysel medzi sebou. V podstate si každý výrobca počítačov musel vyrábať vlastný operačný systém aj aplikácie. To bolo samozrejme drahé, pretože masová výroba a predaj aplikácií nepripadali do úvahy. Predaj softvéru bol obmedzený predajom počítačov, ktoré mali problém presadiť sa v reálnom živote, pretože boli komplikované a použiteľné len na vyhradené účely.

Zjednotením platformy PC sa tento stav náhle zmenil. Aplikácie sa mohli začať predávať vo veľkom. Čím viac aplikácií sa predalo, tým boli ich ceny nižšie, a tým sa zvýšil i dopyt po nových aplikáciách. Prechod na grafické používateľské rozhrania predstavoval ďalší krok, umožňujúci vstup do nového fascinujúceho informačného veku. Jeho následkom sa stali počítače použiteľnejšími, rozšírenejšími, lacnejšími a hlavne prístupnými všetkým vrstvám spoločnosti. Ku koncu deväťdesiatych rokov nastala explózia Internetu. Idea surfovania internetovými stránkami znamenala ďalší rozmach informatiky, výpočtovej a komunikačnej techniky. Počítače sa stali všedným vybavením moderných rodín. Rozšírením softvéru vznikla nutnosť zapodievať sa jeho organizovanou tvorbou. Vznikol pojem softvérový priemysel, a tí čo stáli pri začiatku, sú dnes medzi najlepšimi.

Toto všetko je však len príprava na to, čo nás čaká v najbližšej budúcnosti. Digitálne postupy sa pomaly, ale isto vtierajú medzi bežné aktivity človeka. Hudba je digitálna, fotografia a video sú digitálne, dokumenty sú digitálne. Spôsob ich získavania, spracovania, používania aj zdieľania je tiež digitálny. To všetko poskytuje nevídané možnosti ich použitia na Internete.

Činnosti ako platenie účtov, nakupovanie, komunikácia a zábava sa budú odohrávať čoraz viac prostredníctvom Internetu. Komunikácia na diaľku už nie limitovaná len na zašumený zvuk. Môžeme sa na seba pozerat' naživo, zatiaľ čo sa nachádzame na opačných koncoch zemegule. Keď si vyberáme, do ktorej reštaurácie pôjdeme, pokladáme za samozrejmé, že sa nám zobrazí mapa ako sa tam dostať,

aktuálna ponuka a čašník nám odporučí aj konkrétne jedlo. Toto všetko je možné úplne jednoducho a rýchlo.

Aby to však bolo možné, je potrebné aby vývoj mohol napredovať. Vďaka štandardom a ich masovému rozšíreniu budeme sledovať podobný vývoj, aký sa dal pozorovať u hardvéru v deväťdesiatych rokoch. Úlohu zjednotiteľa v tejto oblasti preberá disciplína web inžinierstva. Už dnes majú mnohí tvorcovia web aplikácií problém s výberom vhodných štandardov z obrovského množstva už existujúcich či novovznikajúcich. Úspešní budú najmä tí, čo vsadia na štandardy a postupy, ktoré prežijú a uplatnia sa v reálnom živote, podobne ako to bolo s hardvérom v rokoch deväťdesiatych.

Aj keď je web inžinierstvo ešte mladá disciplína, priťahuje pozornosť mnohých výskumníkov, vývojárov a iných veľkých hráčov na poli web systémov. Web inžinierstvo sa musí vyvíjať, aby bolo možné efektívne odpovedať na výzvy, ktoré nám web systémy ukladajú a budú ukladať v blízkej i ďalekej budúcnosti. V podstate vzniká nový smer v softvérovom inžinierstve, ktorého úlohou je vyvíjať nové metódy, techniky a prístupy ako odpoveď pre stále náročnejšie a robustnejšie web systémy. Oblasti, ktoré sú v tomto prípade zaujímavé, sú nasledovné:

- analýza požiadaviek a návrh systémov
- modelovanie informácií
- modely procesov
- testovanie, verifikácia a validácia
- výkonnostné testy
- web metrika
- manažment konfigurácie a projektu
- návrh používateľských rozhraní
- personalizácia, orientácia na používateľa
- zabezpečenie kvality
- vzdelávanie

Niektoré z týchto oblastí sú už známe z prostredia vývoja softvéru, tieto sa však od nich jemne odlišujú. A práve preto si zaslúžia väčšiu pozornosť. Z priestorových a časových dôvodov nebolo možné do predkladanej publikácie vtiesnať všetky spomenuté oblasti, ale časť z nich tu je spracovaná. Na web inžinierstve závisí celá

revolúcia v oblasti web aplikácií, ktorá veľkou mierou ovplyvňuje našu spoločnosť. Veľké množstvo aplikácií v oblastiach obchodu, vzdelávania alebo štátnej správy sa presúva práve do web prostredia. A práve preto je potrebné, aby sa rozvinul inžiniersky prístup k vývoju takýchto aplikácií.

Všetci dúfame, že v horizonte niekoľkých rokov sa web inžinierstvo stane vyspelou disciplínou softvérového inžinierstva, ktorá bude akceptovaná a využívaná v celosvetovom meradle ako plnohodnotná veda.

Literatúra

- [AISJ97] C. Alexander, S. Ishikawa, M. Silverstein, and M. Jacobson. *A Pattern Language*. Oxford University Press, New York, 1997.
- [BFIM98] T. Boutell, R. Fielding, U. C. Irvine, and L. Masinter. Uniform Resource Identifiers (URI): Generic syntax. RFC 2396, Internet Society, August 1998.
- [Bou97] T. Boutell. PNG (Portable Network Graphics) specification, version 1.0. RFC 2083, Massachusetts Institute of Technology, March 1997.
- [CL00] L. L. Constantine and L. A. D. Lockwood. *Software for use*. Addison Wesley, 2000. ISBN 0-201-92478-1.
- [GHJV95] Gamma, R. Helm, R. Johnson, and J. Vlissides. *Elements of reusable object-oriented software*. Addison Wesley, 1995.
- [Gin02] A. Ginige. Web engineering: managing the complexity of web systems development. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 721–729. ACM Press, 2002.
- [GM01] A. Ginige and S. Murugesan. Web engineering: An introduction. *IEEE Multimedia*, January-March:14–18, 2001.
- [GPS93] F. Garzotto, P. Paolini, and D. Schwabe. HDM – a model-based approach to hypertext application design. *ACM Trans. Inf. Syst.*, 11(1):1–26, 1993.
- [GRS97] A. Garrido, G. Rossi, and D. Schwabe. Pattern systems for hypermedia, 1997.
- [HBv94] L. Hardman, D. C. A. Bulterman, and G. van Rossum. The Amsterdam hypermedia model: adding time and context to the dexter model. *Commun. ACM*, 37(2):50–62, 1994.

- [HS94] F. Halasz and M. Schwartz. The Dexter hypertext reference model. *Commun. ACM*, 37(2):30–39, 1994.
- [Hyp] Hypermedia design patterns repository. www.designpattern.lu.unisi.ch/index.htm.
- [ISB95] T. Isakowitz, E. A. Stohr, and P. Balasubramanian. RMM: a methodology for structured hypermedia design. *Commun. ACM*, 38(8):34–44, 1995.
- [ISO86] ISO. Information processing – Text and Office Systems – Standard Generalized Markup Language (SGML). ISO 8879, International Organization for Standardization, 1986.
- [ISO93] ISO. Information technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s. ISO/IEC 11172, International Organization for Standardization, 1993.
- [ISO94] ISO. Information technology – Digital Compression and Coding of Continuous-tone Still Images. ISO/IEC 10918, International Organization for Standardization, 1994.
- [ISO97] ISO. Information technology – Virtual Reality Modelling Language (VRML). ISO/IEC 14772-1, International Organization for Standardization, 1997.
- [ISO99] ISO. Information technology – Coding of audio-visual objects. ISO/IEC 14496, International Organization for Standardization, 1999.
- [JBR99] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1999. ISBN 0201571692.
- [JCJÖ92] I. Jacobson, M. Christenson, P. Jonsson, and G. Övergård. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison Wesley, 1992. ISBN 0-201-54435-0.
- [Jef01] R. E. Jeffries. XProgramming.com: an extreme programming resource, 2001. <http://www.xprogramming.com/>.
- [Jel98] I. Jelínek. Od textu k dobrému hypertextu. *Softwarové noviny*, pages 102–106, júl 1998.
- [Kos00] J. Kosek. *XML pro každého*. GRADA, 2000. ISBN 80-7169-860-1.

- [Lan94] D. B. Lange. An object-oriented design method for hypermedia information systems. *The 27th Annual Hawaii International Conference on System Sciences IEEE*, 3, 1994.
- [LH02] P. J. Lynch and S. Horton. *Web Style Guide*. Yale University, 2002. <http://www.webstyleguide.com>.
- [LRS98] F. Lyardet, G. Rossi, and D. Schwabe. Using design patterns in educational multimedia applications, 1998.
- [LRS99] F. Lyardet, G. Rossi, and D. Schwabe. Patterns for adding search capabilities to web information systems, 1999.
- [MK93] H. Maurer and F. Kappe. Structured browsing of hypermedia databases. *VHCI'93.*, 1993.
- [MMC01] E. Mendez, N. Mosley, and S. Counsell. Web metrics-estimating design and authoring effort. *IEEE Multimedia Web Engineering Part 1*, January–March:50–57, 2001.
- [MW01] A. McDonald and R. Welland. Agile Web Engineering (AWE) process, 2001. www.dcs.gla.ac.uk/~andrew/TR-2001-98.pdf.
- [Net04] NetRatings Inc. *Gaming Doubles in Europe in One Year*, February 2004. http://banners.noticiasdot.com/termometro/boletines/docs/audiencias/nie%lsen-netratings/2003/0203/netratings_pr_030219_uk.pdf.
- [Ngu00] H. Q. Nguyen. Testing web-based applications. *Software Testing & Quality Engineering*, May/June:www.stqmagazine.com, 2000.
- [Pre00] R. S. Pressman. What a tangled web we weave. *IEEE Software*, 17 1, January/February:18–21, 2000.
- [proa] <http://www.procmail.org>.
- [prob] Web engineering methodology and development manual. www.e-negociogalicia.com/proxecto/documentacion/Web_Engineering_Methodo%logy_and_Development_Manual.pdf.
- [Rei00] D. J. Reifer. Web development: Estimating quick-to-market software. *IEEE Software*, 17 6, November/December:57–63, 2000.
- [RK02] G. Rossi and N. Koch. Patterns for adaptive web applications. In *Seventh European Conference on Pattern Languages of Programs*, 2002.

- [RLS00] G. Rossi, F. Lyardet, and D. Schwabe. Patterns for E-commerce applications, 2000.
- [rob] <http://www.robotstxt.org/wc/norobots.html>.
- [RSD01] G. Rossi, D. Schwabe, and J. Danculovic. Patterns for personalized web applications. *Proceedings of EuroPlop'01*, pages 423–436, 2001.
- [RSG97] G. Rossi, D. Shwabe, and A. Garrido. Design reuse in hypermedia design applications development. *Proceedings of ACM International Conference on Hypertext (Hypertext'97)*, April 7-11, 1997.
- [RSL98] G. Rossi, D. Schwabe, and F. Lyardet. Patterns for designing navigable information spaces, 1998.
- [RSL00] G. Rossi, D. Schwabe, and F. Lyardet. User interface patterns for hypermedia applications. In *Proceedings of the working conference on Advanced visual interfaces*, pages 136–142. ACM Press, 2000.
- [sal] sales@apttest.com. *Software Testing Resources and Tools*. www.apttest.com.
- [spi] <http://www.w3.org/Search/9605-Indexing-Workshop/ReportOutcomes/Spidering.txt>.
- [SR02] T. J. Shelford and G. A. Remillard. *Real Web Project Management: Case Studies and Best Practices from the Trenches*. Addison-Wesley, 2002.
- [SRB96] D. Schwabe, G. Rossi, and S. D. J. Barbosa. Systematic hypermedia application design with OOHD. In *Proceedings of the the seventh ACM conference on Hypertext*, pages 116–128. ACM Press, 1996.
- [SW98] G. Schneider and J. P. Winters. *Applying Use Cases: A Practical Guide*. Addison Wesley, 1998. ISBN 0-201-30981-5.
- [Tak97] K. Takahashi. Analysis and design of web-based information systems. *Sixth International World Wide Web Conference, HyperNews*, 1997.
- [vBM03] H.M.A. van Beek and S. Mauw. Automatic conformance testing of internet applications. In *Proceedings of the 3rd International Workshop on Formal Approaches to Testing of Software (FATES 2003)*, pages 53–63, Montreal, Canada, October 2003.
- [W3Ca] W3C. Extensible Markup Language (XML) 1.1. <http://www.w3.org/TR/xml11/>.

- [W3Cb] W3C. HTML 4.01 Specification. <http://www.w3.org/TR/html401/>.
- [W3C01] W3C. Scalable Vector Graphics (SVG) 1.0 Specification, 2001. <http://www.w3.org/TR/2001/REC-SVG-20010904/>.
- [WK99] S. Ward and P. Kroll. Building web solutions with the rational unified process: Unifying the creative design process and the software engineering process, 1999. <http://www.rational.com/>.
- [WT02] J. Wack and M. Tracey. *DRATF Guideline on Network Security Testing*. NIST, Gaithersburg, Feb 2002.
- [Žár01] J. Žára. Nejmladší z rodu „empeg“. *CHIP*, pages 143–145, január 2001.
- [ŽBF98] J. Žára, B. Beneš, and P. Felkel. *Moderní počítačová grafika*. Computer Press, 1998. ISBN 80-7226-049-9.