VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TESTOVÁNÍ ROZHRANÍ WEBOVÝCH APLIKACÍ

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE Bc. STANISLAV STUDENÝ
AUTHOR

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TESTOVÁNÍ ROZHRANÍ WEBOVÝCH APLIKACÍ

TESTING OF WEB-BASED USER INTERFACES

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE — Bc. STANISLAV STUDENÝ
AUTHOR

VEDOUCÍ PRÁCE — Mgr. MAREK RYCHLÝ
SUPERVISOR

BRNO 2007

Abstrakt

Semestrální projekt je zaměřen na vývoj nástroje pro testování webového rozhraní. Úvodní kapitoly jsou věnovány vývoji softwarového produktu, kde se čtenář dozví co je životní cyklus software, jaké typy životních cyklů existují a jejich stručnou charakteristiku. V třetí kapitole je čtenář seznámen se základním zařízením pro sítě 802.11 a jeho konfigurací. Testováním webových rozhraní se zabývá následující kapitola, kde jsou uvedeny přístupy, jak lze webová rozhraní testovat. Pátá kapitola popisuje návrh aplikace. V závěru je shrnutí dosavadních výsledků, přínosů a budoucího vývoje aplikace.

Klíčová slova

HTML, Web, XML, Testování, Chyba, Proxy server, Bezdrátové zařízení, Přístupový bod, 802.11, Směrovač, Přepínač, Software

Abstract

This term project is focused on the developement of a tool for testing of the 802.11 web based configuration interfaces. The beginning chapters are about the software lifecycle, its types and the brief characteristics. In the third chapter, the reader is introduced into the 802.11 access point device and its configuration. The next chapter is focused on testing of the web based interfaces and a web testing methodology. The fifth chapter describes the application design. In the last chapter, there is a summary of findings and the future development progress.

Keywords

HTML, Web, XML, Testing, Error, Proxy server, Wireless device, Access Point, 802.11, Router, Switch, Software

Citace

Stanislav Studený: Testování rozhraní webových aplikací, semestrální projekt, Brno, FIT VUT v Brně, 2007

Testování rozhraní webových aplikací

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Mgr. Marka Rychlého.

. . . . . .
Stanislav Studený
3. ledna 2008

Obsah

Kapitola 1

Úvod

V tomto dokumentu jsou postupně vysvětleny základní postupy vývoje informačního systému pro Web tj. specifikace, návrh, testování a vývojové cykly, kterými informační systém projde. Dále se zde budeme zabývat webovým konfiguračním rozhraním zařízeními pro sítě bezdrátové sítě 802.11. V následujících kapitolách této práce jsou uvedeny možné metody testování webového rozhraní a jejich zhodnocení. Na základě hodnocení je pak uveden návrh aplikace pro testování Webových konfiguračních rozhraní na zařízeních sítí 802.11. V závěru této práce jsou zhodnoceny přínosy navrhované aplikace a další postup vývoje.

Kapitola 2

Vývoj softwarového produktu

Vývoj softwarového produktu spadá do jeho životního cyklu. Životní cyklus softwarového produktu lze vymezit od doby vzniku prvního požadavku na něj až po okamžik, kdy skončí jeho používání. Definuje fáze, kroky, metody, nástroje a požadované výstupy softwarového projektu. Životní cyklus software má několik fází. Počet těchto fází bývá různý. Záleží zde na jemnosti rozdělení jednotlivých fázi a jejich provázanosti navzájem mezi sebou.

2.1 Specifikace

Fáze specifikace zahrnuje specifikaci požadavků uživatele pro kterého je software vyvíjen a jejich analýzu. Při řešení je tato část jedna z nejtěžších, neboť zahrunuje zejména komunikaci mezi lidmi tj. mezi vývojářem a zákazníkem. Zákazník ne vždy je schopen zformulovat své požadavky tak, aby nebyly přehnané, v zadaném termínu nerealizovatelné, případně požadavky které jsou si navzájem protichůdné.

2.2 Návrh

Fáze návrhu je část, kdy se provádí popis struktury softwarového projektu, který má být implementován a uveden do provozu. Toto zahrnuje zejména popis dat, rozhraní a komponent, ze kterých se případně bude systém skládat, uživatelského rozhraní a další.

2.3 Testování

Tato fáze zahrnuje oblast, která se zabýva odhalováním chyb. Jedná se v podstatě o dva způsoby testování. Prvním z nich je posuzování kvality implementovaného zdrojového kódu nebo spouštěním spustitelného kódu, podle toho známe-li zdrojový kód či nikoliv. Není-li znám zdrojový kód, pak se při testování vychází ze specifikace funkcí, které daný softwareový produkt nabízí a testuje se na očekávané hodnoty.

2.4 Vývojové cykly

Model vodopád je nejstarším modelem, který prosazuje systematický vývoj software. Jednotlivé fáze vývoje striktně následují postupně za sebou a jednotlivými fázemi se prochází

v zásadě jednou. Uživatel či zákazník je zde pouze u specifikace a analýzy požadavků. Pak čeká až do doby, kdy je systém dokončen a připraven k převzetí.

Iterativní model prochází fázemi opakovaně. Cílem je v každé nové iteraci (vývojovém cyklu) vytvářený systém doplnit o nové vlastnosti nebo vylepšení. Každá iterace je nová verze systému tzv. `build`. Výsledkem iterace by měl být systém, který ale nemusí mít implementovanou veškerou, dle návrhu požadovanou funkčnost. Ta bude postupem implementována a rozšiřována v následujících iteracích.

Kapitola 3

Zařízení pro síť 802.11

Síť 802.11, označovaná také jako Wi-Fi, se stala standardem lokální bezdrátové sítě známé také jako WLAN[1], které vycházejí ze specifikace IEEE 802.11 viz. [10].

Klienti se k bezdrátové síti Wi-Fi mohou připojit dvěma způsoby. Prvním z nich je připojení bez přístupového bodu. Těmto Wi-Fi sítím se říká Ad-hoc sítě. Zde jsou klienti mezi sebou ve vzájemném přímém rádiovém spojení, kde první klient, tuto síť řídí a ostatní klienti se k němu připojují [6].

Druhým typem připojení do bezdrátových sítí Wi-Fi jsou připojení k přístupovému bodu[2]. Klienti sítě pak mezi sebou komunikují prostředníctvím přístupového bodu. Tyto prvky se pak používají pro budování infrastruktury bezdrátových sítí 802.11.

3.1 Přístupový bod

Jak již bylo zmíněno, tyto prvky sítě se využívají k budování infrastruktury, ale existují i jiné aplikace, ve kterých lze schopnosti těchto zařízení uplatnit, např. jako tiskové servery, file servery, směrovače a jiné.

V většině případů se jedná o embedded systémy, tj. malé počítače, architektury ARM[3] [9] s Operačním systémem Linux. Toto umožňuje výrobci velkou flexibilitu ve volbě softwarového vybavení. Mnoho těchto zařízení jsou po stránce hardware skoro identická, ale liší se možnostech nastavení.

Protože v těchto zařízeních běží Operační systém Linux, lze zde spustit i různé služby, které slouží pro provoz, konfiguraci a správu.

3.1.1 Specifikace konfiguračního rozhraní

Důležitou částí při vývoji přístupového bodu je konfigurace. Ta by měla být lehce přístupná, intuitivní a jednoduchá. Proto výrobci těchto zařízení volí způsob konfigurace prvku jeho uživatelem využitím Web rozhraní.

Pomocí webového prohlížeče se uživatel dostane na webovou stránku, kde je možné měnit vlastnosti přístupového bodu. Tato komunikace mezi servrem(přístupový bod) a klientem(uživatel), probíhá promocí protokolu HTTP, viz. [5].

Obecně to bývá pracovní režim, ve kterém přístupový bod pracuje, nastavení WLAN rozhraní, LAN rozhraní, Firewall, NAT, UPnP a další volby.

[1] zkratka - Wireless Local Area Network
[2] anglicky - Access point
[3] zkratka - Advanced RISC Machine

3.1.2 Vývojový cyklus

V první fázi vývoje se prováděl návrh výchozího datového modelu sloužícího k ukládání a načítání dat pro konfiguraci viz. [7]. V další fázi se vyvíjely dvě aplikace sloužící pro konfiguraci operačního systému přístupového bodu a speciální web server, které nad navrženým datovým modelem pracují. Dále byly navrženy šablony webových stránek do nichž web server načítá informace o aktuálním nastavení z interního datového modelu a generuje tak stránky pro webový prohlížeč klienta.

Všechny výše zmiňované fáze se stále vyvíjejí. S nově příchozími požadavky týkajících se rozšíření funkčnosti, oprav chyb nebo optimalizace se vždy provádí vyhodnocení požadavku. Je-li rozhodnuto, že požadavek bude realizován, pak se provádí případné změny jednom i více bodech v závislosti na požadavcích realizace :

1. návrh datového modelu
2. úprava aplikace pro konfiguraci OS
3. úprava funkce webového serveru

Kapitola 4

Testování webového rozhraní

Existuje několik přístupů, jak testovat webové rozhraní. V této kapitole si popíšeme nejdůležitější z nich z hlediska automatizovatelného testování.

4.1 Testování modulů

Způsob, který umožňuje, jak odhalovat chyby, tak velmi nízkou kvalitu kódu, je testování modulů[1]. Každý modul představuje část kódu, která může být i samostatná Web stránka nebo nějaký aplet. Testování modulů ma dva hlavní účely. Prvním je, že každá verze Webové aplikace splňuje nebo naopak předčívá očekávaný stupeň kvality. Druhý se využívá pro odhalování chyb na úrovni kódu už při prvních fázích vývoje ve vývojovém cyklu.

V současné době je většina webových stránek napsána s využitím značkovacího jazyka HTML[2]. Tento jazyk je standardem a jeho specifikaci lze nalézt v [2]. Validace HTML kódu podle specifikace s postupným vývojem webových aplikací a nových standardů zajistí jeho přenositelnost.

Existence HTML standardu umožňuje, aby byly elementy web stránek chápány co největším počtem webových prohlížečů stejně. Zajištění dodržování standardu HTML umožní maximalizovat přístupnost webu pro co největší škálu uživatelů.

Mnohá většina dnešních stránek je generována dynamicky. Existují šablony, do kterých se kód vkládá (generuje) na základě uživatelem zvolených akcí. Šablonu lze otestovat již po její sestavení. Vygenerovanou stránku lze otestovat, ale nelze ji jednoduše testovat tak, aby byly pokryty všechny možnosti generování. Tyto metoda sice existuje, lze nalézt v [8], ale je časově velice náročná jak z hlediska vývojáře, tak testera a obecně se při vývoji webového rozhraní nepoužívá.

4.2 Validace CSS

Kaskádové styly nebo také CSS[3] jsou způsob, jak umožnit vývojářům a uživatelům řídit zobrazování webové stránky. Umožňují přiřadit různé styly pro stejný HTML dokument. Webové prohlížeče používají sérii pravidel, kterými určují pořadí (kaskádu) stylů nad HTML dokumentem.

Prohlížeče typicky využívají následující hiearchii pravidel :

[1] anglicky - Unit Testing
[2] zkratka - HyperText Markup Language
[3] CSS je anglickou zkratkou pro Cascading Style Sheets

1. Atributy HTML tagů mají přednost před všemi styly.
2. Inline styly se používají k dědění na stránce.
3. Interní styly jsou definovány na začátku stránky a jsou aplikovány na celý její obsah.
4. Extení styly jsou definovány mimo stránku. Často jsou definovány v jediném externím souboru, který je pak použit pro celý web. Výsledkem je tak konzistentní vzhled všech stránek webu.
5. Externí linkované styly.
6. Vlastní nastavení webového prohlížeče.
7. Standardní nastavení webového prohlížeče.

Pokud více stylů ve stejné úrovni hiearchie obsahuje konfliktní pravidlo, pak pravidlo, které je prohlížem načteno jako poslední bude upřednostněno. Validace kaskádových stylů je stejně důležitá jako validace HTML. Validuje se struktura stylů, vlasnosti a jejich atributy.

4.3 Kompatibilita

I když je kód stránky striktně podle posledních standardu, není zaručeno, že se zobrazí na všech webových prohlížečích stejně. Vývojáři a testeři musí úzce spolupracovat, aby zajistili, že webová stránka bude kompatibilní s různými prohlížeči tak, aby byla zobrazitelná na všech stejně. Existuje mnoho různých způsobů které mohou ovlivnit prezentaci webové stránky. Zajistit efektivně testování všech možných případů je takřka nemožné a nelze ji úplně zautomatizovat.

4.4 Navigace

Existuje několik druhů odkazů používaných na webových stránkách. `Interní` odkazy vedou na uživatele na další stránku v rámci vlastního webu. `Absolutní` URL poskytují kompletní cestu ke stránce, např. pro OS Linux :

```
/home/WWW/index.html
```

pro OS Windows :

```
C:/adresar/podadresar/index.html
```

`Relativní` URL ukazují na stránku vzhledem k pozici aktuálně zobrazené stránky. Např. `podadresar/index2.html`.

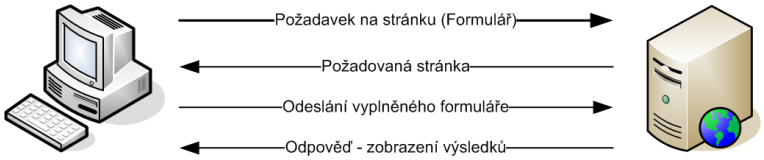
Obecně by měly interní odkazy vždy využívat relativní URL. Webová aplikace je pak méně náchylná k porušeným odkazům, když je web přesunut do jiného prostředí, např. z testovacího do produkčního. Tyto typy chyb lze testovat procházením zdrojového kódu vygenerované HTML stránky a hledání příslušných sekvencí, které začínají např. `C:/` nebo různé další sekvence, kterými by se dala identifikovat absolutní cesta. K těmto účelům lze vhodně použít regulárních výrazů. Hledání však může mít i nějaká falešná pozitiva. Je tedy nutné, aby tester zvážil, zda-li se jedná o chybu, či nikoliv.

`Externí` odkazy se odkazují na externí webové stránky, které jsou mnohdy ve správě jiné organizace a je potřeba zajistit, aby nebyly porušeny. Při změně struktury webu, např. reorganizací souborů do jiné adresářové struktury se tyto linky poruší.

Lze provádět testy odkazů na velká a malá písmena. Některé web servery jsou citlivé na velká a malá písmena v URL. Není-li dáno jinak, je vhodné, aby odkazy obsahovaly pouze malá písmena.

4.5 Interakce uživatele

Interakce web serveru s klientem probíhá např. pomocí forulářů[4]. Ty umožňují uživateli odeslat většinou jakýkoliv typ informace na web server pro okamžité zpracování. Formulář je formátovaná část dokumentu mezi tagy `<form>` a `</form>` obsahující vstupní pole a další elementy, lze nalézt v [1], které mají buď již předvyplněné hodnoty nebo jsou prázdné a uživatel má možnost si je vyplnit. Informace položek závisí na použití formuláře. Jakmile je formulář vyplněn a zaslán na web server, pak po zpracování údajů odešle web server výsledek klientovi. Průběh komunikace je ilustrován na obrázku 4.1.



Obrázek 4.1: Zpracování formuláře Web serverem

Elektronické formuláře jsou využívány především na internetu, protože jazyk HTML má podporu pro zobrazování elementů formuláře jako je textové pole, žaškrtávací pole a další. Pokud nejsou formuláře adekvátně dobře otestovány, mohou způsobit nepředvídatelné a nepříjemné problémy. Například použití funkcí Vpřed a Zpět ve webovém prohlížeči může způsobit při vkládání dat do databáze, že jsou data vloženy/modifikovány více krát apod.

Data se dají odesílají na web server pomocí dvou metod POST a GET. Při posílání dat na web server pomocí metody GET jsou data uložena v URL. Zde by se mělo kontrolovat, zda-li nedošlo k oříznutí v případě většího rozsahu dat přenášených pomocí metody GET, protože velikost URL bývá různě omezena. Metoda POST odesílá data z fromuláře až po odeslání URL a nejsou omezena na svou velikost. Tato metoda je preferovaná pro zasílání dat mezi klientem a serverem.

Validaci položek lze testovat jak na straně serveru, tak na straně klienta JavaScriptem, avšak některé prohlížeče mají možnost JavaScript zakázat nebo ho nepodporují. Většinou to jsou pouze textové prohlížeče např. lynx, links apod. Tímto způsobem lze kontrolu na straně klienta obejít a web server tak může dostat po odeslání formuláře nerelevantní data.

Součásti interakce uživatele s web serverem jsou i Cookies. Jsou to data, která jsou serverem zaslána klientskému prohlížeči a mohou obsahovat skoro cokoliv. Může to být např. unikátní id generované serverem, datum a čas, IP adresa apod. Když klient požaduje stránku např. X po webserveru, pak je mu cookie zaslána spolu s ní. Když klient požaduje jinou stránku Y ze stejného webu, pak je cookie zaslána zpět na web server spolu s požadavkem. Typický účel je identifikace návštěvníků, když se vracejí zpět na Web. Další informace o cookie lze nalézt v [3].

[4] anglicky - Forms

Testování cookies může probíhat na změnu kdy je parametr v :

- přidán
- smazán
- zaměněn
- změna jednoho či více parametrů na neplatnou hodnotu

4.6 Shrnutí

Z hlediska testování na odhalování chyb zavlečených nově implementovanými vlastnostmi je základní metodou testování reakce Webového rozhraní na uživatelský vstup s kombinací testování modulů, tedy na sledování změn Web rozhraní na základě interakce uživatele na různých stránkách nastavení. Při vývoji stačí sestavit případy testů (scénáře) ve formě očekávaných nebo neočekávaných vstupů a očekávaných výstupů a aplikovat je. Pak lze lehce sledovat změny na Web rozhraní.

Dalším vhodným způsobem je testování validity HTML kódu a jeho struktury. A to zejména v případech, kdy se stránky testují čistě jako HTML šablony nebo až po vygenerování, jsou-li dynamické. Snadno se může během implementaci stát, že vývojář při generování přidá pro některý z elementů nepatřičný atribut nebo dojde k vygenerování stejné stránky různě. Tímto jsou myšleny různé odchylky v kódu mezi řádky.

Kapitola 5

Návrh aplikace

5.1 Specifikace

Aplikace pro testování Webového rozhraní by měla být univerzální, multi-platformní a přenositelná, protože lze očekávat, že se testuje na různých Operačních systémech. K vývoji bude využita C++ knihovna Qt verze 4 viz. [11] a bude probíhat iterativním životním cyklem, kdy každou další iterací bude doplněna funkčnost a provedena případná oprava implementačních chyb předchozí verze.

Dále bude využito jazyka XML pro popis textovacích skriptů. Tento jazyk je určen zejména pro výměnu dat mezi aplikacemi a umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí. Další informace o jazyku XML lze nalézt v [4].

Aplikace se bude skládat z následujících částí :

- Proxy server
- XML[1] generátor testovacích skriptů
- Nástroj interpretaci a vyhodnocování testovacích skriptů

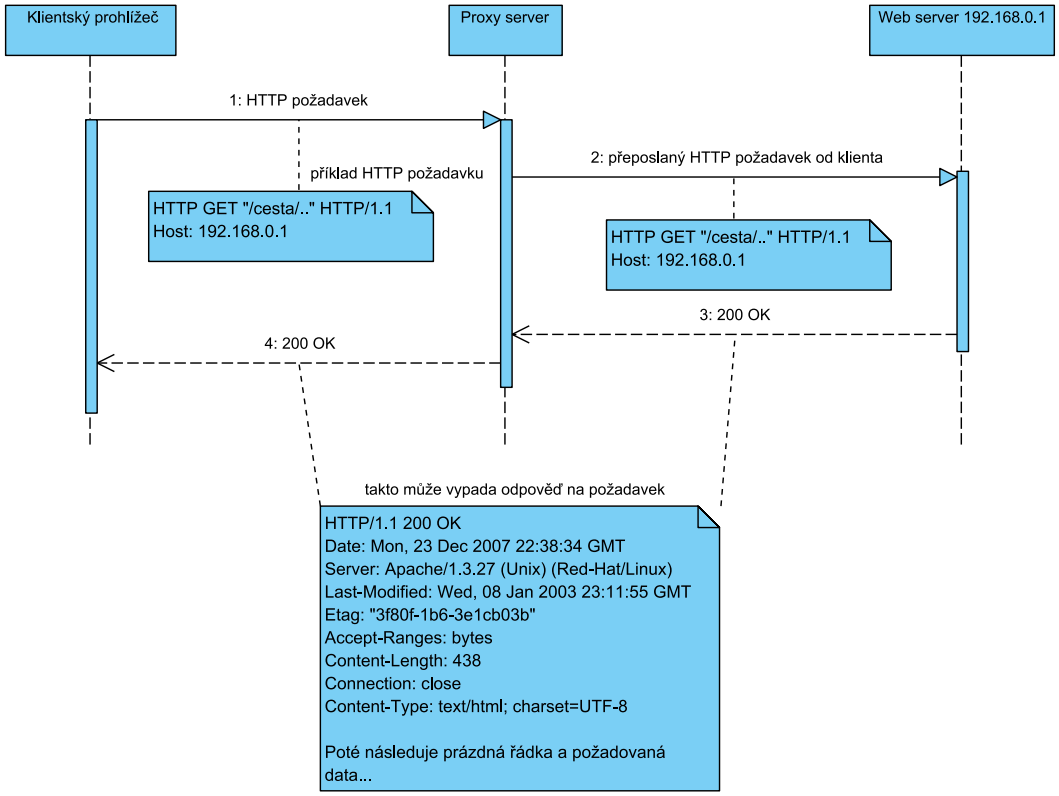
5.1.1 Proxy server

Protože se bude jednat o testování pomocí užitavelské interkace s Web rozhraním, bude pořeba zajistit odposlouchávání komunikace mezi Web serverem a jeho klientem. Pro odposlech komunikace bude využit proxy server, na který bude Webový prohlížeč přesměrován. Proxy-server bude zároveň sloužit i jako nástroj pro sběr komunikačních dat pro testovací software. Jako klient se bude předpokládat Webový prohlížeč s podporou proxy serveru, na který pak bude veškerá komunikace přesměrována. Tento postup zajistí univerzalitu. Příklad komunikace klienta s Web serverem je veden na obrázku 5.1.

5.1.2 XML generátor testovacích skriptů

XML generátor bude sloužit k vygenerování testovací skript buď pro načtenou stránku v klientovi, která byla přenesena a zachycena na proxy serveru nebo šablonu otevřenou přímo generátorem. Generování skriptu bude probíhat přehledně pomocí průvodce[2] v několika

[1] zkratka - The Extensible Markup Language, česky rozšiřitelný značkovací jazyk
[2] anglicky - Wizard



Obrázek 5.1: Komunikace klienta s Web serverm přes transparentní proxy server

krocích. Spuštění generátoru bude možné z nástroje pro vyhodnocování a interpretaci testovacích skriptů, ale i samostatně z příkazové řádky s parametry, aby bylo možné vytvářet testovací skripty automatizovaně.

Formát testovacího XML skriptu bude vypadat následovně :

```
<?xml version="1.0" encoding="utf8"?>
<Webtest>
  <Created>01.01.2008 20:36:35</Created>
  <Creator>Web Tester</Creator>
  <Purpose></Purpose>
  <Comment></Comment>
  <ChangeLog>
    <!-- id pořadí modifikace, název, datum, kdo modifikuje -->
    <Change id="" name="" date="" modifier="">
      <!-- popis změny -->
    </Change>
  </ChangeLog>

  <!-- testování může skládat z několika kroků -->
  <Step id="" url="" skip="[0|1]">

    <Data>
```

```
      <!-- očekávaný datový vstup -->
      <InputFiles>
        <InputFile id=""></InputFile>
      </InputFiles>
    </Data>

    <Tests>

      <!-- Test formuláře -->
      <FormTest formId="" submitURL="" execute="[0|1]">
        <Inputs method="GET">
          <Input name="" size="" type="[input|textarea|select]" regex="">
            <!-- hodnota vstupu -->
          </Input>
        </Inputs>
        <Inputs method="POST">
          <Input name="" size="" type="[input|textarea|select]" regex="">
            <!-- hodnota vstupu -->
          </Input>
        </Inputs>
      </FormTest>

      <!-- Test cookies -->
      <CookiesTest execute="[0|1]">
        <Cookie id="" name="">
          <Parameter name=""></Parameter>
        <Cookie>
      </CookiesTest>

      <!-- Test validity HTML  -->
      <!-- 1 = Ano aktivovat test, 0 = Ne -->
      <HTMLValidityTest execute="[0|1]"></HTMLValidityTest>

      <!-- Test odkazů -->
      <!-- 1 = Ano aktivovat test, 0 = Ne -->
      <HTMLLinksTest execute="[0|1]"></HTMLLinksTest>

      <!-- Test rozdílů -->
      <!-- 1 = Ano aktivovat test, 0 = Ne -->
      <HTMLDiff execute="[0|1]"><HTMLDiff>

    </Tests>

  </Step>

</Webtest>
```

Protože vytváření testů může mít nastarosti i několik vývojářů i testerů, je vhodné, aby byla možnost vést případnou identifikaci testu, jeho účel a také identifikaci úprav v testovacím skriptu. Běžným častějším komentováním změn přímo v testovacím skriptu by vedlo k pozdější ztrátě přehlednosti. Proto je do skriptu sekce mezi značkami `<ChangeLog>` a `</ChangeLog>`, která realizuje výše uvedené. Každá změna je identifikována svým jednoznačným ID a obsahuje datum změny, kdo změnu provedl a také popis změny.

Při testování se může snadno stát, že bude potřeba testovat více stránek, ale ve stejném logickém celku např. skupinu webových stránek týkající se nastavení firewallu. Z tohoto důvodu by bylo vhodné zavést možnosti testování ve více krocích. K tomuto účelu slouží sekce mezi značkami `<Step>` a `</Step>`. Někdy by bylo vhodné mít možnost jak přeskočit některé kroky, protože jsou již třeba otestované. K tomu slouží atribut `skip`. Ten může nabývat hodnot 0 s významem přeskočit krok nebo 1 nepřeskočit.

V každém kroku lze rovněž provádět několik druhů testů, dle možností poskytovaných aplikací. Tyto testy jsou uvedeny mezi značkami `<Tests>` a `</Tests>` pro každý krok a dle vlastního uvážení lze zvolit, zda-li se bude vybraný test spouštět, podobně jako tomu je u jednotlivých kroků.

Protože jsou některé metody testování založené na porovnávání změn před a po uživatelské interakci, je potřeba mít šablonu, podle které se pak porovnávají změny. To je zajištěno vstupní datovou sekcí `<Data>` každého kroku. Do této sekce se uvádí cesta k datovému souboru.

5.1.3 Nástroj pro interpretaci a vyhodnocování testů

Tento nástroj bude realizován jako grafické uživatelské rozhraní[3]. Zde bude mít tester možnost pracovat s testovacími skripty, v reálném čase sledovat komunikaci s Web serverem, včetně detailního výpisu, která se vytváří na základě jeho interakce s klientem. Sběr dat bude probíhat dle diagramu 5.4.

5.1.4 1. Iterace

V první iteraci bude implementováno základní GUI testovací aplikace, Proxy server a jeho ovládání pomocí GUI testovací aplikace. Poté bude prováděno testování Proxi serveru.

5.1.5 2. Iterace

Implementace datové komunikace a sbírání dat testovací aplikací z Proxy serveru. Dále doplnění funkčnosti aplikace o testování a vyhodnocování testů. Na konci této iterace budou prováděny rozsáhlé testy funkčnosti a odstraňování implementačních chyb.

5.1.6 3. Iterace

V třetí fázi bude implementován XML generátor testovacích skriptů, ukádání komunikace mezi klientem a Web serverem, bude přidána možnost testovat web rozhraní pomocí předpřipravených XML skriptů jak z GUI, tak i z příkazové řádky. Dále bude prováděno ladění a odstraňování možných chyb.

[3] angl. zkratka - GUI - Graphical User Interface

5.1.7 Další iterace

Bude provedena lokalizace programu, bude vytvořen systému nápovědy a oprava chyb, které nebyly odhaleny během vývoje a testování.

5.2 Časový plán

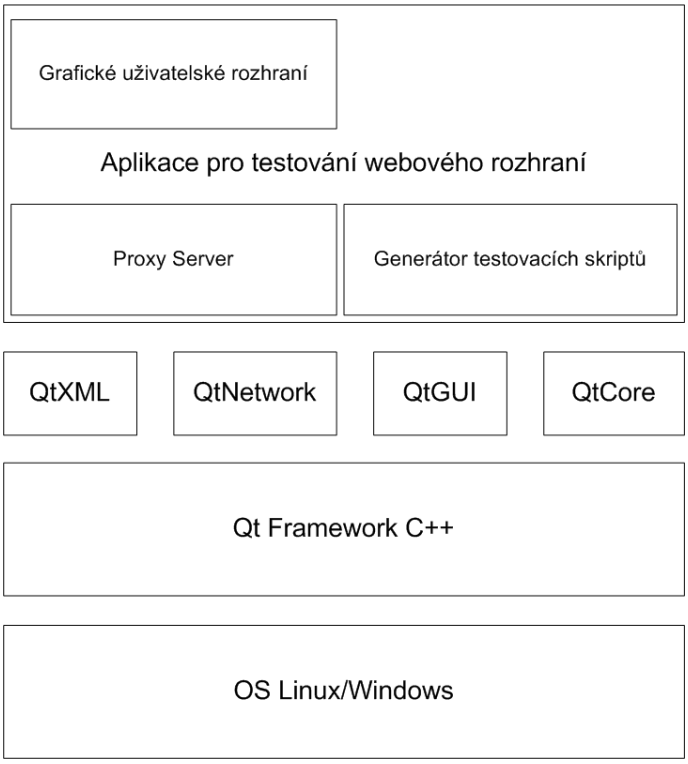
Rozložení časového plánu práce lze nalézt v tabulce 5.1.

48. týden 2007 - 5. týden 2008	1. Iterace
6. týden 2008 - 12. týden 2008	2. Iterace
13. týden 2008 - 17. týden 2008	3. Iterace
18. týden 2008 -	- ostatní

Tabulka 5.1: Tabulka plánu práce

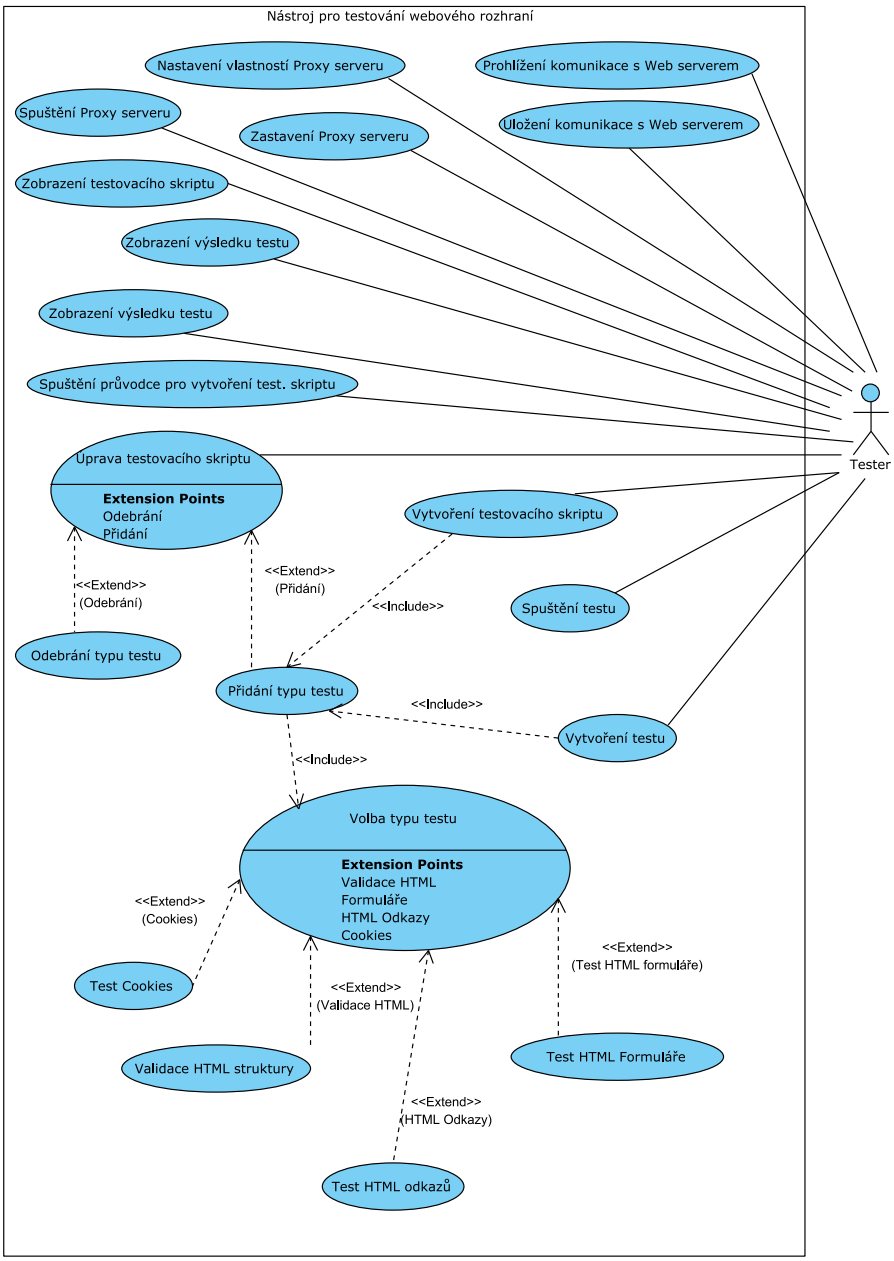
5.3 Návrhové diagramy UML

Na diagramu 5.2 je architektura aplikace. Zde je vidět platformová nezávislost, která je důsledkem využití multiplatformního frameworku Qt.

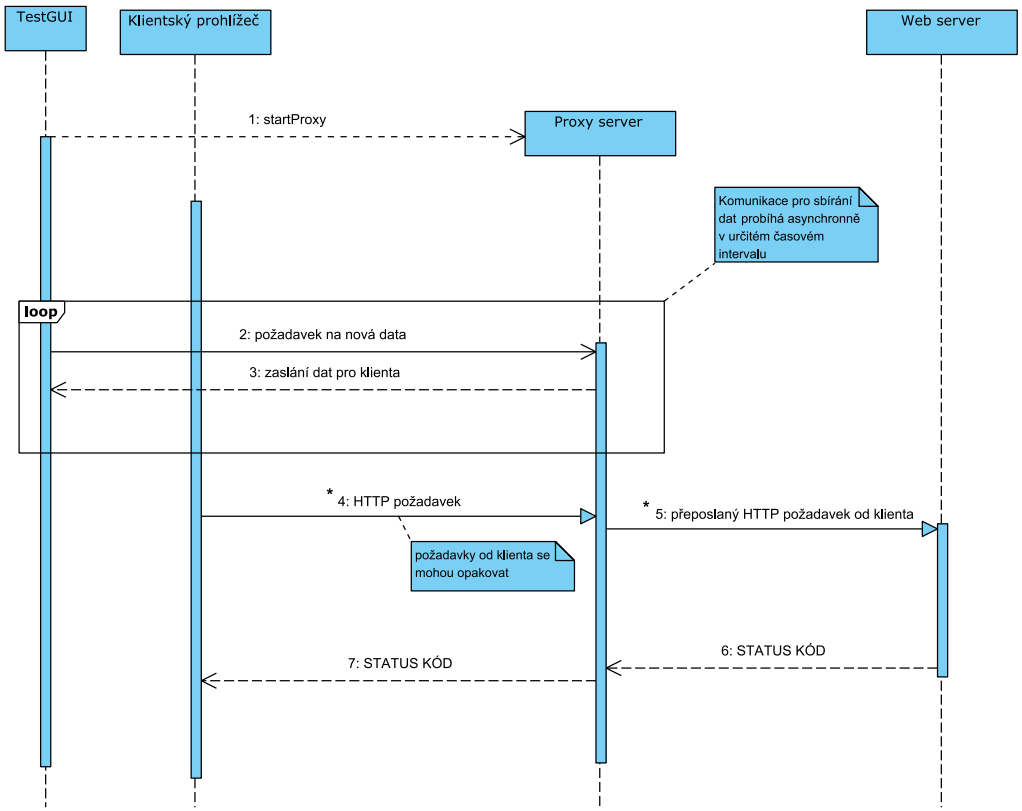


Obrázek 5.2: Architektura aplikace

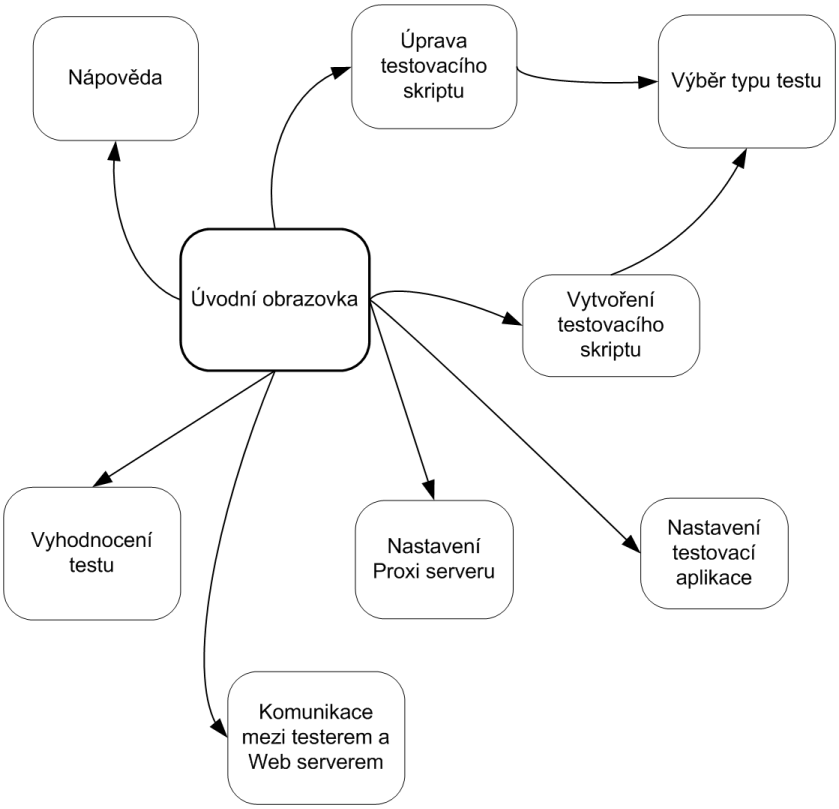
Diagram 5.3 znázorňuje možná použití aplikace testerem. Princip sběru dat z proxy serveru, který je prováděn asynchronně v cyklu, je znázorněn na sekvenčním diagramu 5.4. Diagram 5.5 reprezentuje jednotlivé návaznosti mezi obrazovkama grafického uživatelského rozhraní.



Obrázek 5.3: Diagram případů použití



Obrázek 5.4: Diagram sběru dat testovacím rozhraním



Obrázek 5.5: Diagram návaznosti obrazovek

Kapitola 6

Závěr

Tento návrh aplikace umožňuje částečnou automatizaci testovacích procesů webového rozhraní, která umožní identifikovat případné chyby vzniklé při implementaci. Díky automatickému vyhodnocování testů umožní jejich rychlejší a snažší identifikaci. Další vývoj vzhledem k Diplomové práci bude orientován na dopnění a upřesnění návrhu, implementaci, testování a postupné nasazení k použití.

Literatura

[1] World Wide Web Consortium. Forms. http://www.w3.org/TR/html401/interact/forms.html.

[2] World Wide Web Consortium. HTML 4.01 Specification. http://www.w3.org/TR/html401/.

[3] L. Montulli D. Kristol. RFC2965: HTTP State Management Mechanism. http://www.ietf.org/rfc/rfc2965.txt, říjen 2000.

[4] Elliote Rusty Harold. *XML 1.1 Bible, 3rd Edition*. Wiley Publishing Inc., 2004. ISBN 0-7645-4986-3.

[5] IETF: The Internet Engineering Task Force. RFC2616: Hypertext transfer protocol - HTTP/1.1. http://www.ietf.org/rfc/rfc2616.txt, červen 1999.

[6] Leonard Kleinrock. An internet vision: the invisible global infrastructure. *The Ad Hoc Networks*, 1:3–11, 2003. ISSN 1570-8705.

[7] ANF Data spol. s.r.o. Porta project documentation.

[8] Stefan P. Jaskiel Steven Splaine. *The Web Testing Handbook*. STQE Publishing, 2001. ISBN 0-9704363-0-0.

[9] WWW stránky. ARM architecture. http://en.wikipedia.org/wiki/ARM_architecture.

[10] WWW stránky. Přehled doplňků standardu IEEE 802.11. http://access.feld.cvut.cz/view.php?nazevclanku=&cisloclanku=2005113002. ISSN 1214-9675.

[11] TROLLTECH. Qt: Cross-platform rich client development framework. http://trolltech.com/products/qt.