České vysoké učení technické v Praze
Fakulta elektrotechnická
katedra řídící techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: David Felgr

Studijní program: Otevřená informatika
Obor: Počítačové inženýrství

Název tématu: Metody automatizace testů síťových prvků

Pokyny pro vypracování:
1. S využitím postupů testování podle modelů navrhněte metodiku automatizovaného testování prvků síťové infrastruktury.
2. Navrženou metodiku prakticky implementujte a ověřte její funkčnost.
3. Vaše výsledky porovnejte s dosud používanými postupy.

Seznam odborné literatury:

Vedoucí: doc. Ing. Jiří Novák, Ph.D.


L.S.
prof. Ing. Michael Šebek, DrSc., vedoucí katedry
prof. Ing. Pavel Ripka, CSc., dékan

V Praze dne 10. 3. 2015
diplomová práce

Metody automatizace testů síťových prvků

Bc. David Felgr

Květen 2015

Ing. Pavel Kopec ký, doc. Ing. Jiří Novák, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra Řídicí techniky
Poděkování


Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.
Abstrakt

Cílem této práce je zavést automatizované testování síťových zařízeních společnosti Conel. Zaměřil jsem se zejména na využití testování založeného na modelech v implementaci testovacího systému. Zvolený problém jsem vyřešil návrhem testovacího serveru skládajícího se z několika samostatných částí. Testovací systém se skládá z aplikace obsluhující testování, testovacího API, testovacích skriptů a webové aplikace. Podstata tohoto řešení je rozdělení jednotlivých úkonů do samostatných částí. Vytvořil jsem řešení, které je funkční a lehce upravitelné s přibývajícími požadavky na testování, či na vlastní systém. Přínosem této práce je zdokonalení a zrychlení testování síťových výrobků společnosti Conel.

Klíčová slova

Systémové testování, testování založené na modelech, embedded zařízení, automatizované testování.
Abstrakt

The target of this thesis is introduce automated testing of network devices to Conel company. I’ve focused on the use of models base testing in the implementation of the test system. I solved select problem by proposal testing server. Server composed of several separate parts. the test system consists of testing the application, testing API, test scripts and Web applications. the essence of this solution is the allocation of individual tasks into separate parts. I created a solution that is functional and easily editable with advancing testing requirements, or with new functionality of own system. the contribution of this work is to improve and speed up testing of networking products Conel company.

Keywords

System testing, model based testing, embedded device, automated testing.
Obsah

1 Úvod ............................................. 1
  1.1 Praktické využití .......................... 1
  1.2 Aktuálnost ................................ 2
  1.3 Výstup práce ................................. 2
  1.4 Struktura práce .............................. 3

2 Používané metody testování ................... 4
  2.1 Úrovně testování ........................... 4
      2.1.1 Testování programátorem (Developer testing) 4
      2.1.2 Testování jednotek (Unit testing) ............. 5
      2.1.3 Integrační testování (Integration testing) .... 5
      2.1.4 Systémové testování (System testing) .......... 6
      2.1.5 Akceptační testování (Acceptance testing) ... 7
  2.2 Testovací procesy ........................... 7
      2.2.1 Manuální testování ..................... 7
      2.2.2 Automatizované testování ................. 8
      2.2.3 Testování založené na modelech ............. 9
  2.3 Typy testování ............................. 11
      2.3.1 Testy splnění a selhání .................. 11
      2.3.2 Progresní a regresní testy ............... 11
      2.3.3 Smoke testy ................................ 11
      2.3.4 Funkční a nefunkční testy ............... 11
      2.3.5 Testování bílé a černé skříňky .......... 11
      2.3.6 Statické a dynamické testy ............... 12

3 Dostupné testovací nástroje .................... 13
  3.1 Jenkins CI .................................. 13
  3.2 Testlink ..................................... 14
  3.3 Selenium ..................................... 14
  3.4 VectorCAST .................................. 15
  3.5 Mavernyx ..................................... 16
  3.6 Robot Framework ............................. 17
  3.7 Embedded Unit ................................ 17
  3.8 Linux Test Project ........................... 17
  3.9 Ostatní aplikace ............................ 18

4 Návrh testovací laboratoře ..................... 19
  4.1 Testovací server ............................ 19
      4.1.1 Hardwarová výbava testovacího serveru .... 19
      4.1.2 Softwarová výbava testovacího serveru ...... 19
  4.2 Konfigurovatelné switche .................... 20
  4.3 Testovaná zařízení .......................... 20
      4.3.1 Řada routerů v0 ......................... 21
      4.3.2 Řada routerů v1 ......................... 21
      4.3.3 Řada routerů v2 ......................... 21
      4.3.4 Řada routerů v3 ......................... 22
5 Pohled na modelové testování
5.1 Funkce zařízení 25
5.2 Testovací procedury 26
5.3 Model zařízení 26
5.4 Příklad modelu zařízení 27
5.5 Výhody modelového přístupu 27

6 Testovací program 29
6.1 Adresářová struktura testovacího systému 29
6.2 Struktura databáze 30
6.2.1 Tabulka fvreleases 30
6.2.2 Tabulka platforms 31
6.2.3 Tabulka products 31
6.2.4 Tabulka checkout 32
6.2.5 Tabulka builds platform 32
6.2.6 Tabulka build product 32
6.2.7 Tabulka routers 33
6.2.8 Tabulka functions 33
6.2.9 Tabulka dependences 34
6.2.10 Tabulka routers has functions 34
6.2.11 Tabulka procedures 34
6.2.12 Tabulka tests router 35
6.2.13 Tabulka tests function 35
6.2.14 Tabulka tests procedure 36
6.2.15 Tabulka logs 36
6.3 Popis programu 37
6.3.1 Checkout 38
6.3.2 Compile 38
6.3.3 Remote server 39
   Telnet 40
   SSH 41
6.3.4 Test 41
6.3.5 Clear 42

7 Testovací API 43
7.1 Knihovna cpipe 43
7.2 Knihovna database 43
7.3 Knihovna utils 44
8 Uživatelský interface

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Sekce build</td>
<td>48</td>
</tr>
<tr>
<td>8.1.1 Stránka List releases</td>
<td>48</td>
</tr>
<tr>
<td>8.1.2 Stránka Build platforms</td>
<td>49</td>
</tr>
<tr>
<td>8.1.3 Stránka Build products</td>
<td>49</td>
</tr>
<tr>
<td>8.2 Sekce test</td>
<td>50</td>
</tr>
<tr>
<td>8.2.1 Stránka Devices</td>
<td>50</td>
</tr>
<tr>
<td>8.2.2 Stránka Functions</td>
<td>50</td>
</tr>
<tr>
<td>8.2.3 Stránka Procedures</td>
<td>51</td>
</tr>
<tr>
<td>8.2.4 Stránka Logs</td>
<td>51</td>
</tr>
</tbody>
</table>

9 Návrh testů pro Conel routery

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1 Checkout</td>
<td>52</td>
</tr>
<tr>
<td>9.2 Compile</td>
<td>52</td>
</tr>
<tr>
<td>9.3 Clean</td>
<td>53</td>
</tr>
<tr>
<td>9.4 Funkce firmware</td>
<td>53</td>
</tr>
<tr>
<td>9.4.1 Procedura upload</td>
<td>53</td>
</tr>
<tr>
<td>9.4.2 Procedura start</td>
<td>53</td>
</tr>
<tr>
<td>9.4.3 Procedura check</td>
<td>54</td>
</tr>
<tr>
<td>9.5 Funkce configuration</td>
<td>54</td>
</tr>
<tr>
<td>9.6 Funkce connect</td>
<td>54</td>
</tr>
<tr>
<td>9.6.1 Procedura telnet</td>
<td>54</td>
</tr>
<tr>
<td>9.7 Funkce connect ssh</td>
<td>54</td>
</tr>
<tr>
<td>9.7.1 Procedura ssh</td>
<td>55</td>
</tr>
<tr>
<td>9.8 Funkce mobile</td>
<td>55</td>
</tr>
<tr>
<td>9.8.1 Procedura connect</td>
<td>55</td>
</tr>
<tr>
<td>9.8.2 Procedura ping</td>
<td>55</td>
</tr>
<tr>
<td>9.8.3 Procedura apn</td>
<td>56</td>
</tr>
<tr>
<td>9.8.4 Procedura address</td>
<td>56</td>
</tr>
<tr>
<td>9.8.5 Procedura operator</td>
<td>56</td>
</tr>
<tr>
<td>9.8.6 Procedura mtu</td>
<td>56</td>
</tr>
<tr>
<td>9.9 Funkce mobile edge</td>
<td>57</td>
</tr>
<tr>
<td>9.9.1 Procedura type edge</td>
<td>57</td>
</tr>
<tr>
<td>9.10 Funkce mobile units</td>
<td>57</td>
</tr>
<tr>
<td>9.10.1 Procedura type units</td>
<td>57</td>
</tr>
</tbody>
</table>
9.11 Funkce mobile lte ................................................. 58
  9.11.1 Procedura type lte ......................................... 58
9.12 Funkce mobile ppp ............................................. 58
  9.12.1 Procedura chap ............................................. 58
  9.12.2 Procedura pap ............................................. 58
  9.12.3 Procedura number ......................................... 59

10 Testovací laboratoř v praxi .................................. 60

11 Návrhy na budoucí rozšíření .................................. 63

12 Závěr ................................................................. 64

Literatura ............................................................... 66
Zkratky

IP Internetový protokol
LAN Lokální síť
SSH Zabezpečený shell
API Rozhraní pro programování aplikací
TCP Protokol transportní vrstvy OSI modelu
MBT Testování založené na modelech
CR Návrat na začátek řádku
LF Nový řádek
PID Identifikační číslo procesu
MySQL Databázový systém
Bash Implementace Unixového shellu
EDGE Vylepšené přenos dat pro GSM
UMTS Universální mobilní telekomunikační systém
LTE Vysokorychlostní internet
3G Mobilní technologie třetí generace
CDMA Mobilní komunikační kanál
PPP Point to point protokol
PAP Ověřovací protokol založený na hesle
CHAP Ověřovací protokol založený na výměně klíčů
AT Protokol komunikace s mobilními moduly
GIT Systém správy verzí
1 Úvod

Tématem této diplomové práce jsou metody automatizace testů síťových prvků. Nejdříve popiši definici testování a různé způsoby, jak je možné testování provádět. Definice testování je podle IEEE Software Engineering Body of Knowledge (SWEBOK) následující:

"Softwareové testování se skládá z dynamického ověřování chování programu proti očekávanému chování programu na konečné množině testovacích případů vhodně vybraných z obvykle nekonečné množiny případů."


1.1 Praktické využití

1 Úvod

Obrázek 1 Příklad routeru

Na základě těchto skutečností byl vynesen požadavek na testovací systém, pomocí kterého bude možné automaticky testovat aktuálně vyvíjený firmware na všech vybírěných modelech routerů a jejich volitelných portech. Zvolena byla kombinace integračního testování a testování založeného na modelech. Testování bude pokrývat pouze integrační a systémové testování, jelikož z větší části je firmware testovaných výrobků tvořen opensource programy. Psaní unit testů pro každý open source program je časově a složitostně nepříznivé. Systémové testování by mělo probíhat alespoň jednou denně, aby bylo možno případné chyby odhalit již během vývoje firmware a tím se usnadnil a zrychlil samotný vývoj. Dalším velkým přínosem je zkušenost samotného testování, jelikož testování mohou vymyslet nové testovací procedury a situací namísto opakovaného manuálního testování stejných procedur.

1.2 Aktuálnost

Testování každého výrobku před uvedením na trh, či testování nového firmware před jeho vydáním, je velmi důležitá součást vývoje a neměla by se opomíjet. Hlavním důvodem průběžného testování komplexní funkcionality zařízení je dobré jméno u zákazníka, který nemá zájem o výroba plný chyb. Dalším důvodem průběžného testování je méně práce při pozdějším opravování způsobených chyb. Při zvyšování počtu výrobků a funkcionálu je nutné zvyšovat počet testérů nebo změnit přístup k způsobu testování. Jelikož první řešení se zda být na první pohled neefektivní, tak se v této práci výdají druhou cestou. Při zvyšování efektivity testů používají automatizované testování. Dále pokud to bude možné, a alespoň trochu efektivní, využijí dnes velmi moderní metody testování založeného na modelech. Každému zařízení by měl být vytvořen model, pomocí kterého budou na určených zařízeních spouštěny vybrané testy s danými parametry. V dnešní době, kdy je nutné více a více reportovat, nebo shromažďovat testovací procedury, může automatizovaný systém pomoci ještě více.

1.3 Výstup práce

Hlavním cílem práce je hotové řešení automatizovaného systémového testování všech modelů bezdrátových routerů společnosti Conel. Výstupem řešení bude testovací laboratoř obsahující všechny výrobky, pomocné síťové prvky a testovací server. Dalším výstupem bude aplikace spouštění testů a případnou režii, interface pro zobrazování reportů ze všech testů a možnost administrace testovacího systému. Poslední cíl je vytvoření testovacích procedur pro testování jednotlivých funkcí bezdrátových routerů.
1.4 Struktura práce

Celou diplomovou práci lze rozdělit do třech hlavních částí. V první teoretické části budou rozebírány všechny různé metody a přístupy k testování samotnému, aby bylo možné dále stavět na teoretickém základu. Dále v teoretické části budou prozkoumány všechny možné dostupné produkty určené k samotnému testování, či produkty sloužící k jednotlivým úkonom potřebným pro testování daných produktů. Zde bude kladena snaha využít co nejvíce kvalitních hotových řešení sloužících k účelům testování výrobků společnosti Conel. Ideální cesta by byla nalézt produkt sloužící k našemu účelu, ale jelikož je požadavek velmi specifický, s velkou pravděpodobností bude potřeba z velké části testovací systém vyvinout. Druhá část práce se bude zabývat praktickým návrhem všech částí zabývající se testováním každého routeru. V první fázi bude navrhnuta testovací laboratoř z hlediska potřebného hardwaru. Dále bude popsán návrh a implementace programu zajišťující samotné testování a úkony s testováním související. Následuje kapitola věnující se uživatelskému interfacu pro reportování výsledků a administraci samotného testování. Samostatná kapitola popisuje API pro snadné dopisování nových testovacích procedur. Základní API bude dodáno s testovacím programem a dále bude popsána možnost dopisování nových specifických programů. Stážová část je kapitola popisující testovací procedury jednotlivých funkcionalit. Poslední část je praktická implementace testovací laboratoře a testování celého systému. Výsledky z tohoto testování budou použity v poslední kapitole zabývající se možností budoucího vylepšení a rozšíření testovacího systému.
2 Používané metody testování

V této kapitole se pokusím popsat co nejvíce známých pohledů a způsobů na testování, aby bylo dále možné vybrat, aplikovat a navrhovat testovací systémy s ohledem na dnešní metody a trendy v testování. Nejdříve popišu úrovně testování, kterými by měl každý výrobek před uvedením na trh projít. Dále popiši způsoby testování, kterými může testování proběhnout. V poslední kapitole jsou popsány další nezařazené možné pohledy a přístupy k testování.

2.1 Úrovně testování

Testování výrobků před uvedením na trh prochází několika stupni testování. Některé stupně jsou při vývoji používány bez toho, aby si to vývojáři uvědomili a jiné důležité stupně testování jsou zase často opomenuty. Mním rozebíraný model má celkem 5 stupně testování. Jednotlivé stupně dále popišu a rozeberu jejich přínos a možnosti použití v mém testovacím systému.

![Schéma testovacího modelu](image)

2.1.1 Testování programátorem (Developer testing)

První a úplně nezbytnou fázi testování by měli provádět programátoři. Programátor by si měl zkontrolovat jestli je možné firmware přeložit, a dále jestli jeho nová či opravená funkčionalita funguje správně. V další fázi testování by měl zkontrolovat kód jiný programátor, který kód nepsal. Tuto fázi většinou provádí architekt při zařazování nové či upravené funkce do hlavní větvě repositáře projektu. Všechny chyby odchycené v této fázi testování ušetří spoustu času stráveném v dalších fázích testování.
Testování programátem může vypadat jako samozřejmá věc, která by nemusela být ani uváděna. Bohužel opak je pravdou a i tato situace může nastat. V případě že, je tato fáze vynechána, je pravděpodobné, že spousta chyb je odhalena až ve fázi systémového testování, kdy zjišťování, reportování a oprava chyb stojí značnou režii.

2.1.2 Testování jednotek (Unit testing)

Úroveň testování jednotek obsahuje testování jednotlivých částí nebo modulů software. Za jednotku, neboť část, lze považovat objekt, program či modul. Tato úroveň testování testuje správnost zdrojového kódu a ne funkci programu jako celku. Velmi známým příkladem jsou JUnit testy v jávě, kde je ke každé třídě a metodě vytvářena testovací třída či metoda.

Unit testy je výhodné použít při tvorbě nového projektu, jelikož s unit testy je potřeba počítat již při návrhu zdrojového kódu a při návrhu kódu zároveň testy vytvářet. Dopisování testů do již existujícího projektu by stálo neúrodnou náhlu a mnoho úprav kódu pro přizpůsobování samotného projektu pro unit testování.

Zdrojový kód pro výrobky testované navrhovaným testovacím systémem je vyvíjen již 10 let a zároveň na těchto zdrojových částech jsou stavěny i nové výrobky. Navíc přes devadesát procent firmwareu používá opensource řešení. Díky těmto dvěma skutečnostem je v nynější situaci nerealné přidat unit testování do navrhovaného testovacího schématu pro routery firmy Conel.

2.1.3 Integrační testování (Integration testing)

Po předchozích dvou úrovních testování, jež jsou prováděny programátory, přichází fáze, kdy se hotový výrobek dostává do ruky testerům. Testěři většinou provádí dvě úrovňové testování, integrační testování a systémové testování. Někdy jsou tyto dvě fáze spojovány do jedné a nazývají se systémové integrační testování. Obě úrovňové budou dále detailněji popsány.

Obrázek 3 Příklad grafického znázornění integračního testování

Pomocí integračního testování testujeme integraci jednotlivých komponent mezi ostatní komponenty, dále také integraci jednotlivých komponent do operačního systému či na konkrétní hardware. Jako příklad mohu uvést testování integrace programu odesílání sms zpráv na operační systém Linux, který běží na hardwaru konkrétního
Příklad znázorňuje, že není testováno pouze odeslání SMS zpráv, ale komponenta v závislosti na operačním systému a hardwaru. Jednotlivé komponenty mohou být například subsystémy, databázové implementace, infrastruktura, rozhraní a systémové konfigurace. Integrační testování lze z testování vypustit, jelikož chyby nalezené v této fázi by byly odhaleny ve fázi systémového testování.

Integrační testy navrhují testéři na základě čtyř základních skutečností, a to softwarový a systémový design výrobku, architektura firmwaru, pracovní postup s danou komponentou a možnými případy použití. Na základě téhoto čtyř skutečností tester navrhuje testovací případy a postupy. Podle téhoto postupů jsou jednotlivé komponenty dále testovány manuálně testery či automaticky automatem.

Testovací automat bude v prvním kroku testovat integračními testy všechny základní komponenty routeru, jako například posílání SMS zpráv, běžícím na každém z 50 různých výrobků. Zde je nejlépe vidět přínos testovacího automatu. Veskutečnosti by tester měl provést test integrace všech funkcionalit na všech padesáti odlišných výrobcích, což je časově značně náročné. Zatímco automat tento test může provést každý den na všech výrobcích paralelně během krátké doby. Tímto testováním je ověřena integrace daného programu na všech výrobcích a neunikne žádná chyba způsobená chybou v firmwaru, či chybu některé ze součástí routeru, čímž může být například odlišná implementace odeslání SMS v bezdrátovém modulu.

### 2.1.4 Systémové testování (System testing)


![Obrázek 4: Příklad systémového testování](image)

Systémové testy mohou obsahovat funkční i nefunkční testy, které jsou dále popsané v sekci věnované těmto typům testování. Dále je možné testovat kvalitu či rychlost přenosu dat, které mohou být prováděny na výrobcích ve standartním, ale i ve stíženém prostředí, například v klimatické komoře nebo v EMC prostředí. Na systémové testování je také možné pohlížet jako na testování bílé či černé skřínky. Oba způsoby jsou také dále popsány v kapitole věnující se dalším typům testování.

V navrhnoutém případě testování bude systémové testování prováděno ve stejném kroku a pomocí stejných nástrojů jako integrační testování, čili tento model se blíží dříve
zmíněné možnosti spojení systémových a integračních testů. V navrhovaném případě testování lze někdy určit hranice mezi systémovými a integračními testy a někdy je toto rozdělení obtížné určit. Většina systémových testů, obdobně jako integrační testy, bude možné provádět automaticky pomocí testovacího automatu. Jiné systémové testy jako například testy v klimatické komoře, budou muset být dále z kapacitních důvodů prováděny manuálně, jelikož všech padesát výrobků se do klimatické komory nevejdou. Naopak tyto testy nezávisí na změně firmwaru, čili ve většině případech stačí pouze jedno provedení klimatických testů při vyvinutí nového hardwaru výrobku a ne při každé změně firmwaru.

2.1.5 Akceptační testování (Acceptance testing)


2.2 Testovací procesy

Fáze systémového a integračního testování lze provádět pomocí třech různých přístupů k testování. Všechny tři přístupy se od sebe liší hlavně v délce a složitosti návrhu testování a v délce samotného provádění testů. V neposlední řadě se liší jejich pokrytí testovacích případů a tím i kvalita testování. Podle složitosti návrhu testů by bylo možné popisovat přístupy testování seřadit sestupně na testování založené na modelech, automatizovaném testování a manuálním testování. Cílem této práce je automatizovat, a tím zkrátit provádění testů, a zároveň rozšířit možnosti testování. Z cíle práce tedy vyplývá, že se budu snažit pokusit o přechod z nynějšího manuálního testování na automatizované testování a v nejlepším případě na testování založené na modelech.

2.2.1 Manuální testování


Druhou fází manuálního testování je samotné provádění testů. Testy se provádějí manuálně přímo na testovaném objektu podle předepsaných testovacích procedur. Provedení testů je opakováno při každé změně v firmwaru výrobku a dle jeho komplexnosti bývá i velice časově náročný. Z toho důvodu bývají některé testy vypouštěny na úkor kvality testování celého výrobku. Samotné testování je práce manuální dle předepsaných pokynů a také velmi často se opakuje, z toho plyne, že tuto práci může vykonávat tester bez znalostí návrhu testování samotných výrobků a jejich technologií. Dále se tento proces přímo nabízí k nějakému zlepšení jakoukoliv automatizací.
2 Používané metody testování

Obrázek 5 Schéma manuálního testování [1]

Ve společnosti, kde bude testovací systém nasazován, se nyní všechny testy prováđejí manuálně dle předepsaných testovacích procedur. Jak už bylo v úvodu zmíněno, při rychle rostoucím počtu výrobků a jejich funkcionalit je nadále tento systém testování neúdržitelný. Provádění integračních a systémových testů se budou snážit přesunout do jednoho ze sofistikovanějších způsobů testování. Dále pro specifické testy, jako například EMC testy nebo klimatické testy v teplotní komoře, nebude z kapacitních důvodů možné plně automatizovat a bude možné navrhnout moduly pro zjednodušení manuálního testování.

Mohou nastávat i případy manuálního testování, kdy nejsou vytvořeny testovací plány a testovací procedury, kde je možné později doložit adekvátní výsledky testování. Další nevýhodou tohoto přístupu k testování je vynechaní velkého množství testovacích případů, jelikož se jedná spíše o náhodné testování, proto tento postup není určitě doporučován.

2.2.2 Automatické testování

Druhým, a sofistikovanějším způsobem provádění testů, je automatizované testování, někdy také nazývané testování založené na skriptech. Automatické testování lze rozdělit do tří základních fází. První fáze vytvoření testovacího plánu je shodná s manuálním testováním.

Druhou fází automatického testování je implementace testovacích procedur do spustitelných skriptů. Skriptovací testy mohou být napsány v nějakém standardním programovacím jazyce nebo v jazyku přímo určeném k psaní testovacích skriptů. V našem systému bude k psaní testovacích skriptů použit skriptovací jazyk Bash a jednoduché programy napsané v programovací jazyk C. V automatizovaném přístupu testování nám přibyla další role programátoru potřebného k implementaci testovacích procedur do spustitelných skriptů. Testovací skript je spustitelný skript nebo program, který provede jednu testovací proceduru. Testovací skript obvykle obsahuje inicializaci.
2.2 Testovací procesy

Testovací zařízení, uvedení testovacího zařízení do požadovaného kontextu, vytvoření vstupních testovacích hodnot, předání vstupních hodnot do testovaného zařízení, nahrání odpovědi od testovaného zařízení, nakonec porovnání odpovědi a očekávaného výstupu a vyhodnocení výsledku.


Naopak tento přístup přínáší větší režii při změně funkionality či přidání nového zařízení. Pokud je změněna testovací procedura či přímo funkcionalita výrobku, musí být předčleny a přidány testovací skripty. Tato údržba může být v případech intenzivního vývoje stejně časově nákladná jako tvorba nových testovacích procedur pro danou funkcionalitu.

2.2.3 Testování založené na modelech

Nejsofistikovanějším řešením testování je testování založené na modelech, známé taky jako MBT (Model Base Testing). Zjednodušeně lze model tohoto systému popsat následovně. Tester vytvoří model testovaného zařízení, z tohoto modelu se automaticky vygenerují testovací skripty, které jsou pak spouštěny nad testovaným výrobkem. U tohoto testování odpadá spoustu času při návrhu a úpravách testovacích skriptů. Naproti tomu je velmi časově náročné navrhnout samotného testovacího systému a modelu testovaného výrobku. Samozřejmě, že všechno není tak jednoduché, jak se na první pohled zdá, a tak dále jsou detailně popsány všechny fáze tohoto způsobu testování. Jednotlivé fáze MBT jsou vývoj modelu testovaného zařízení, generování abstraktních testů z modelu, převedení abstraktních testů na spustitelné testy, spuštění testů na testovaném
zařízení a analýza výsledků testů.

Obrázek 7 Schéma testování založeného na modelech [1]

Prvním krokem testování založeném na modelech je tedy vytvoření abstraktního modelu testovaného zařízení. Model by měl být jednodušší nežli samotné zařízení a měl by se zaměřit na jeho klíčové vlastnosti.

Druhým krokem testování založeném na modelech je generování abstraktních testů z hotového modelu. Jelikož by ve většině případů bylo vygenerováno nekonečné množství testovacích případů, tak je potřeba určit nějaká testovací kritéria, aby bylo možné vygenerovat konečné množství testů. Tyto testy jsou sekvenční operací nad modelem. Používaní jednodušený pohled na testované zařízení a nejsou přímo spustitelné.


Ve čtvrtém kroku jsou spouštěny konkrétní testy na testovaném systému. Tato fáze je shodná s třetí fázi automatizovaného testování. Tedy může používat stejný systém a ve zjednodušené variantě lze říci, že jde pouze o jiné generování testovacích skriptů. Dále jde tento krok rozdělit na online a offline testování. Při offline testování se generují testy při každém spuštění testu. Při offline testování jsou testy předem generovány a pokud nenastane změna v testovacím modelu, jsou používány stejné již předem generované testy.

V posledním pátém kroku se analyzují výsledky spuštěných testů a jejich korektní chování. V případě neúspěšného kroku se analyzuje přičina a místo vzniku chyby. Nejčastější místa vzniku chyby jsou chybný model, chybný adaptér kódu a v neposlední řadě může chyba vzniknout chybou funkcí testovaného výrobku.

Z popisu tohoto způsobu testování je vidět, že v případě správné implementace tohoto systému na testovaný produkt by mohlo výrazně usnadnit práci při testování. Samotná
implementace je velmi složitá a ne všechny testované objekty lze efektivně popsat tímto systémem. Nasazení systému testování založeného na modelech zrostkota po několika letech i ve velkých společnostech jako například IBM. V testovacím systému pro výrobky společnosti Conel bude snaha použít systém testování založeného na modelech alespoň na nevyšší abstraktivní úrovni. [2]

2.3 Typy testování

Poslední podkapitola typy testů probírá pojmy z testování, které nebyly obsaženy v žádném z předchozích modelů a v testování se občas používají či naopak je některý z předchozí modelů používá a nebyly detailně popsány.

2.3.1 Testy splněním a selháním

Testy splněním používají pro vstupní data pouze množinu dat, které testovaný systém musí správně vyhodnotit. Taktéž se chováme k systému korektním způsobem, při tom kontrolujeme jestli odpověď od testovaného systému se shoduje s očekávanou odpovědí. Naopak při testech selháním zacházíme s testovaným systémem nekorektně a na vstup mu přivádíme data, které systém neumí vyhodnotit a kontrolujeme jestli systém nespadl nebo systém nevračí odpověď shodující se s očekávaným výstupem.

2.3.2 Progresní a regresní testy


2.3.3 Smoke testy

Smoke testy je označení testů obsahující pouze jednoduché testování spustitelnosti produktu a jeho základních funkcionalit. Většinou se toto testování provádí před sys-témovými testy. Největší význam těchto testů přichází ve výrobě nových produktů pro ověření funkčnosti vyrobeného výrobku.

2.3.4 Funkční a nefunkční testy

Pomocí funkčních testů jsou testovány všechny funkce implementované v testovaném výrobku a jejich správné fungování. Tyto testy jsou popisovány v předchozích kapitolách. Další metodou jsou často opomíjené nefunkční testy testující funkce výrobku přímo nesouvisící s jeho funkcionalitou. Jedná se například o testování výkonu celého výrobku nebo jeho části. Kontroluje se zde jestli výrobek dosahuje požadovaného výkonu a zároveň jestli je při zvýšené zátěži stále dobře funkční.

2.3.5 Testování bílé a černé skřínky

Testování bílé skřínky je prováděno, pokud při navrhování testů má tester přístup a využívá zdrojových kódů testovaného výrobku. Naopak při testování černé skřínky tester zdrojové kódy k dispozici nemá a k návrhu testů využívá pouze dokumentace k danému výrobku.
2 Používané metody testování

2.3.6 Statické a dynamické testy

Statické testy nepotřebují ke svému provádění spuštěný program. Naopak dynamické testy ke svému fungování potřebují spuštěný funkční program. [3]
3 Dostupné testovací nástroje

V kapitole dostupné testovací nástroje popíšu zkoumanou část dostupných open-source i komerčních nástrojů pro jednotlivé fáze testování. U všech testovacích nástrojů bude posuzován přínos v případě nasazení v testovacím systému pro výrobky společnosti Conel.

3.1 Jenkins CI

Jenkins CI je nástroj pro kontinuální integraci. Kontinuální integrace spočívá v čas- tém kompilování zdrojového kódu a následném spouštění testů nad tímto softwarem. Jenkins CI je nástroj napsaný v programovacím jazyce Java. Tento nástroj má podporu kompilace různých projektů díky veliké možnosti nastavení a velké podpoře v podobě rozšiřujících pluginů, jejichž počet neustále roste.

Obrázek 8 Logo produktu http://www.jenkins-ci.org

Jenkins CI je dobrou a přehledným buildovacím systém s podporou mnoha programova- cích jazyků. Velikou výhodou tohoto systému je opensource řešení projektu, naopak nevýhodou jsou velké systémové požadavky nástroje. Tento systém sice podporuje tes- tování, ale podporuje pouze testování na úrovni testování jednotek, které v navrženém systému nebude vůbec využito. Jenkins CI naopak nepodporuje funkční a systémové testování na konkrétním hardwaru, které je v testovacím systému vyžadováno.

Obrázek 9 Příklad otevřené aplikace Jenkinesis

Použití nástroje Jenkins CI pro kontinuální integraci v testovacím zařízení bylo možné pouze ve fázi buildování všech firmwarů. Skripty zajišťující správnou funkci s bu- ildovacím programem Llib je potřeba napsat stejně jak při použití systému Jenkins, tak při vlastním spouštění překladu. LTIIB je image systém používající se pro překlad firm- waru většiny testovaných výrobků. Taktéž samotné spouštění těchto skriptů bude v tes- tovacím systému již implementováno v části obsluhující spouštění testovacích skriptů. Na základě těchto skutečností jsem upřednostnil použití vlastního spouštění kompi- lačních skriptů, hlavně z důvodu jednotného systému pro vizualizaci všech informací o průběhu všech fází překladu a testování.
3 Dostupné testovací nástroje

3.2 Testlink

Testlink je nástroj pro správu a organizaci testů. Nástroj je vyvinutý jako webová aplikace napsaná v jazyce PHP využívající databázi MySQL nebo PostgeSQL. Aplikace je zdarma i pro komerční účely, jelikož je šířená pod licencí GPL.

Obrázek 10  Logo produktu http://www.testlink.org


Obrázek 11  Příklad otevřené aplikace Testlink

Nástroj Testlink může být účinný v případě manuálního testování pro snadnou správu testů a reportování výsledků téhoto testů. I v případě navrženého testovacího systému společnosti Conel by bylo možné systém nasadit. Jestliže by byl systém nasazen, byly by nutné úpravy tohoto systému k požadavkům automatického testování a speciálním požadavkům na modulárnost testovaných výrobků. Po krátkém zkoumání jsem zjistil, že úpravy pro automatické testování a úpravy k potřebám hierarchie testovací laboratoře by byly rozsáhlé. Dále systém Testlink působí velmi nepřehledně. Díky témtě závěrům jsem se rozhodl pro nepoužití nástroje Testlink pro spravování testů a reportování výsledků.

3.3 Selenium

na každý prohlížeč. Poslední Selenium Grid umožňuje paralelní spouštění testů na více strojích i počítačích.

Obrázek 12 Logo produktu http://www.seleniumhq.org


3.4 VectorCAST

Nejlepším zkoumaným testovacím nástrojem byl komerční produkt VectorCAST od společnosti VECTOR Software. VectorCAST je platforma přímo určená k testování embedded zařízení podporující velkou škálu zařízení a testů.

Obrázek 13 Logo produktu http://www.vectorcast.com/

Samotný produkt je rozdělen do několika částí, které jsou dodávány zvlášť podle potřeb zákazníků a typu cílové testované aplikace. První dva programy se zabývají testováním jednotek a integračním testováním. První program je VextorCAST/Ada, který je určen pro testování softwaru napsaného v jazyce Ada. Ada je robustní staticky typovaný programovací jazyk určený pro programování kritických aplikací. Tento jazyk v testovaných zařízeních nikdy nebyl použit a tak se dále tímto programem nebudu zabývat. Druhým programem určeným pro testování jednotek a integrační testování je VectorCAST/C++ určeným pro aplikace psané v jazycích C a C++.


Pro lepší správu regresního testování slouží nástroj VectorCAST/Manage. Nástroj je vhodný pro vývoj aplikací s dlouhým životním cyklem, či s většími vývojovými týmy. Členové celého týmu mohou pohodlně spouštět regresní testy a sledovat jejich výsledky. Nástroj dále obsahuje podporu testování aplikací na více operačních systémech, více cílových platformách a různých konfiguracích. Testování stejného modelového případu ve všech možných situacích je důležité pro stoprocentní ověření systému. Nástroj umož-
3 Dostupné testovací nástroje

Obrázek 14 Příklad otevřené aplikace VectorCast

řešuje také funkci testování založené na změnách, kdy nástroj sleduje změny zdrojových kódů a spouští pouze testy modulů, na kterých byly provedeny změny.

VectorCAST/Cover analyzuje využití kódu v průběhu systémového testování. Pomocí tohoto nástroje je možné zjistit, jaká část aplikace nebyla při samotném systémovém testování testována. Druhým nástrojem pro analýzu kódu VectorCAST/MCDC můžeme provést analýzu všech možných stavů a podstavů, kam se aplikace psaná v jazyce C či C++ může dostat.


Řešení pro testování embedded aplikací VectorCAST, a jeho všechny doplňující balíky, je velmi silným nástrojem. Po detailní zkoumání tohoto nástroje a jeho funkci onáli nemohu ani tento nástroj doporučit pro účely automatické testování výrobků společnosti Conel. Hlavním důvodem nevadností použití tohoto nástroje je potřeba zdrojového kódu k tvorbě jak testování jednotek, tak integračního testování. Firmware v testovaných výrobkách je z devadesáti procent tvořenopensource projektu. Tvorba takových testů na veškerém software by byla velmi dlouhá práce s nejasným výsledkem, taktéž projekt neobsahuje žádnou podporu systémového testování a návrh testů vzhledem k modulárnosti testovaných zařízení.

3.5 Maveryx

Maveryx je komerční řešení pro automatizované testování aplikací s grafickým uživatelským rozhraním. Tato aplikace podporuje testování aplikací napsaných v jazyce Java a aplikací pro operační systém Android. Aplikace Maveryx se nebude hodit k použití v testovacím systému, ani pro řešení žádného mezikroku testování, jelikož žádná z podporovaných technologií není v testovaných zařízeních použita.
3.6 Robot Framework

Robot Framework je framework pro automatizované akceptační testování a vývoj řízený testy. Základní testovací možnosti lze rozšířit pomocí nových knihoven napsaných v jazyce Java nebo Python, pomocí kterých je možné doplňovat nová klíčová slova.

Robotic Framework je pěkný příklad, jak by mohl testovací framework vypadat. Specifické požadavky testovacího frameworku pro testování síťových embedded zařízení tento framework neobsahuje. Samozřejmě by se daly všechny knihovny dopsat pomocí přidavných knihoven a systém upravit k potřebám testovacího zařízení, jednodušší cestou však bude vizit dobré zkušenosti z testování tohoto frameworku a postavit vlastní API vhodné pro testování embedded aplikací.

Robot Framework je opensource projekt pod Apache Licencí 2.0. Robot Framework je nezávislý na operačním systému a aplikacích, jelikož jádro frameworku používá Python.

3.7 Embedded Unit

Embedded Unit je nástroj pro testování na úrovni testování jednotek. Nástroj je určen pro testování softwaru pro embedded aplikace. Jelikož tuto úroveň testování nebudu prozatím implementovat do testovacího systému, tak nebude tento nástroj prozatím využit.

3.8 Linux Test Project

Linux Test Project má za cíl testování stability, spolehlivosti a robustnosti Linuxového jádra a souvisejících funkcí. Tímto nástrojem bychom mohli testovat Linuxovou distribuci pro architekturu testovaných zařízení. S takto detailními testy se pro testovací laboratoř prozatím nepočítá, ale pro kvalitní testování výrobků se bude do budoucna hodit.
3 Dostupné testovací nástroje

3.9 Ostatní aplikace

API pro vytváření jednotlivých testů bude využívat různé, převážně opensource programy ze světa síťových technologií, například Curl pro komunikaci mnoha síťovými protokoly. Všechny tyto použité programy budou popsány v kapitole věnující se API testovacího systému.
4 Návrh testovací laboratoře

Pro testování všech výrobků společnosti Conel nejdříve navrhně strukturu testovací laboratoře. Testovací laboratoř bude ethernetová síť obsahující všechny testované výrobky společnosti Conel, různá příslušenství připojené k jednotlivým výrobkům, testovací server a konfigurovatelné switche. Fyzicky bude pro testovací síť vyroben stojan s obdělníkovým půdorysem asi 60cm na 80cm a výškou jeden a půl metru. Konfigurovatelné switche a testovací server budou umístěny ve spodní části stojanu. Na dvou protilehlých stěnách stojanu budou v patrech namontovány DIN lišty, na které se budou přidělovat testované routery, napájecí zdroj či různá další zařízení potřebná k testování interfaců testovaných zařízení.

4.1 Testovací server

Jádrem testovací laboratoře bude server, na kterém poběží všechny aplikace zajišťující testování a reportování výsledků. Testovací server je připojen ke všem zařízením v testovací síti a odděluje testovaná zařízení od fiřemní sítě. Testovací server může být fyzicky umístěn jinde nežli uvnitř racku, protože ho s testovací síti propojují pouze dva ethernetové kabely.

Obrázek 16 Ubuntu server

4.1.1 Hardwarová výbava testovacího serveru

Testovací server bude počítač s parametry procesor Intel core i7, 16GB RAM a 256GB SSD diskom. V případě potřeby budou pro archivování starších logů a přeložených firmware doplněny další klasické magnetické disky. Pro účely testování bude server osazen dvěma gigabitovými ethernetovými kartami pro komunikaci s testovanými výrobky a jednou gigabitovou ethernetovou kartou pro připojení serveru do fiřemní sítě. Server dále má osazeno jedno WiFi rozhraní pro testování WiFi konektivity testovaných výrobků.

4.1.2 Softwareová výbava testovacího serveru

Operační systém testovacího serveru jsem zvolil Linuxovou distribuci Ubuntu, jelikož operační systém Ubuntu je používán jako referenční systém při vývoji ve společnosti, kde bude testovací systém nasazován. Nasazena bude verze Ubuntu Server 14.04.1 LTS
s prodlouženou podporou na pět let. Na serveru bude umístěna databáze uchovávající všechny informace o testování a webová aplikace zobrazující výsledky z testování. Databázový systém použitý k uchovávání výsledků bude asi jeden z nejznámějších systémů a to MySQL. Pro fungování webové aplikace bude na server nainstalován Apache server s PHP. Spouštění testů obsluhuje testovací aplikace testlab, která je umístěna na testovacím serveru. Na server se také instaluje řada malých samostatných programů tvořících testovací API určené pro zjednodušení psaní jednotlivých testů. Programy instalované na server, které jsou využívány programem testlab a testovacím API budou popsány v samostatných kapitolách u programů, kde jsou použity. Na serveru bude dále umístěna kopie vzdáleného repozitáře pro rychlejší stahování zdrojových kódů testovaných projektů při každém spuštění testování. V dalších fázích vývoje testovacího zařízení by na tomto serveru mohly být testovány systémy R-SeeNet pro monitorování routerů a SmartCluster pro jednoduchou správu VPN tunelů.

4.2 Konfigurovatelné switchy


Obrázek 17 Zapojení testovacího serveru a switchů

4.3 Testovaná zařízení

Bezdrátové routery jsou zařízení, které tvoří přes devadesát procent aktuálního sortimentu firmy Conel a právě tato zařízení budou v první fázi testovány v testovací laboratoři. Bezdrátové routery tvoří dnes celkem čtyři modelové řady, které dále obsahují jednotlivé výrobky podle technologie bezdrátového připojení a úrovni výbavy možnosti rozhraní. Všechny zařízení mají možnost přidělení na DIN lištu a tímto způsobem budou upevňeny na testovací stojan.
4.3 Testovaná zařízení

4.3.1 Řada routerů v0

Takzvaná nultá řada routerů obsahuje pouze dva výrobky. Prvním výrobkem je ER75i, disponující EDGE technologií, jedním ethernet rozhraním a možností osazení jednoho volitelného portu. Druhým výrobkem této řady je UR5i, který se liší pouze bezdrátovou technologií, místo EDGE technologie disponuje rychlejší UMTS technologií. Oba tyto výrobky jsou postaveny na operačním systému uClinux a i přes jejich stáří se firmware stále udržuje. U těchto modelů bude připojeno pouze ethernet rozhraní, které bude připojeno do konfigurovatelného switche. Testování volitelného portu bude popsané v samostatné kapitole věnující se testování konkrétních volitelných portů. Jinými rozhraními tato nultá řada routerů nedisponuje.

4.3.2 Řada routerů v1

Řada routeru v1 není od nulté řady marketingově ani koncepčně nijak oddělena. Řada je z hlediska vyvoje oddělena, protože je založena na původně konceptním operačním systému Linux, namísto uClinux použitém u příchozích výrobků. Řada obsahuje výrobky UR5i disponující HSPA+ technologií a LAN routerem XR5i, který nedisponuje žádnou bezdrátovou technologií.

a) ER75i v plastové krabici.  

b) ER75i v kovové krabici.

Obrázek 18  Příklad vzhledu řady routerů a v1.

4.3.3 Řada routerů v2


a) UR5i v2 v kovové krabici.  

b) LR77 v2 v plastové krabici.

Obrázek 19  Příklad vzhledu řady routerů v2.
4.3.4 Řada routerů v3


![Spectre v3 LTE v plastové krabici](image1.png)  ![Spectre v3 LTE v kovové krabici.](image2.png)

Obrázek 20 Příklad vzhledu řady routerů v3.

4.4 Volitelné porty

Každý router lze osadit alespoň jedním volitelným portem. Jednotlivé volitelné porty rozšiřují router o další rozhraní. Testování musí být každý volitelný port alespoň v každé modelové řadě routerů. Popis volitelných portů a způsob zapojení při testování je popsán v následujících kapitolách týkajících se jednotlivých volitelných portů.

4.4.1 Ethernet 10/100

Volitelný port Ethernet 10/100 rozšiřuje router o další fyzické Ethernet rozhraní. Tento port bude testován stejně způsobem jako základní ethernet na routeru. Všechny tyto porty budou připojeny do konfigurovatelného switche a pomocí VLAN budou odděleny od sítě, kde jsou zapojeny primární Ethernet porty všech zařízení.

4.4.2 RS232

Volitelný port RS232 rozšiřuje router o standardní sériové rozhraní. Sériová rozhraní budou testována trojím způsobem. V prvním případě budou propojeny dva routery se sériovým rozhraním a mezi nimi budou přenašena data. Druhým způsobem testů bude připojení loopback deskys, která umí na sériovém rozhraní odpovídat na pár základních příkazů protokolu AT. Například na příkaz ATI loopback deska odpovídá název testovaného rozhraní a hlášku OK. Poslední testovací případ sériového rozhraní bude připojení a komunikace testovaného zařízení a libovolného zařízení jiného výrobce. [6]

4.4.3 RS485/RS422

Volitelný port RS485/RS422 rozšiřuje router o přepínatelné sériové rozhraní RS485 a RS422. Testování těchto portů je identické s rozhraním RS232. Testovat bude potřeba obě tato rozhraní. Výběr jednoho z rozhraní je prováděno pomocí jumperů na desce volitelného portu.
4.4.4 M-BUS master

Volitelný port M-BUS master rozšiřuje router o další typ sériového rozhraní. Testování tohoto portu bude prováděno zapojením M-Bus slave loopback destičky, která umí odpovídat na základní dotaz vyčtení zařízení pomocí M-BUS protokolu. Druhým způsobem testování bude zapojení reálného M-Bus měřiče.

4.4.5 I/O module CNT

Volitelný port I/O module CNT rozšiřuje router o další binární vstupy, binární výstup, čítačové vstupy a analogové vstupy. Testování tohoto portu bude prováděno pouze jedním způsobem. Volitelný port CNT se bude testovat pomocí loopback desky CNT, která podle nastavení binárního vstupu postupně dle známého postupu nastavuje svoje binární a analogové výstupy.

4.4.6 WiFi

Volitelný port WiFi rozšiřuje router o možnost připojení se do WiFi sítě. Tento rozšiřující port lze osadit pouze do modelové řady v2. Řada routerů v1 nepodporuje WiFi rozhraní. Řada routerů v3 má možnost osadit WiFi rozhraní již na základní desce. WiFi rozhraní se bude testovat proti WiFi připojení testovacího serveru, WiFi připojení CISCO routeru a v neposlední řadě proti jinému Conel routeru s podporou WiFi.

4.4.7 SDH

Volitelný port SDH rozšiřuje možnost routeru o připojení SD karty. Kompatibilita portu SDH je stejná jakou u portu WiFi. Testování portu SDH bude pomocí zápisu a přečtení dat z karty umístěné v držáku SDH portu. Žádná konektivita v případě tohoto portu nebude potřeba.

4.4.8 Switch

Volitelný port Switch rozšiřuje router o další tři switchované ethernet porty. Po připojení portu Switch do routeru je první ethernet switchován do všech tří portů routeru u modelové řady v2. Modelová řada router v3 po připojení volitelného portu switch obsahuje 3 nezávislé ethernet rozhraní switchované do tří portů.

4.5 Cisco router

Další důležitou součástí testovací sítě je bezdrátový Cisco router. Router je zapojen do testovací sítě pomocí etherretu. Router dále využívá mobilní a WiFi připojení. Cisco router je součástí testovací sítě z důvodu testování vybraných protokolů oproti jinému referenčnímu routeru.

4.6 Další pomocné přístroje

V testovací sítě budou dále přibývat různá zařízení pro možnosti lepšího testování jednotlivých rozhraní všech routerů. Mezi tyto zařízení budou například patřit různé měřiče se sériovým rozhraním RS232 nebo M-BUS měřiče. Dále budou v testovací sítě umístěny konkurenční výrobky pro ověření funkčnosti testovaných výrobků s jinými síťovými prvky.
Obrázek 21  Schéma testovací laboratoře
5 Pohled na modelové testování


![Obrázek 22 Pohled na modelové testování](image)

5.1 Funkce zařízení


Funkce definuje vlastnosti chování dané funkcionality zařízení, způsob testování a předpokládaný výsledek testu na modelu. Dále je možné definovat hierarchii funkcí, aby nebylo možné testovat nějakou funkci bez předchozího úspěšného testu jiné funkce. Na příklad nemá smysl testovat nahráni nového firmwaru do výrobku, pokud se nepovede úspěšně přeložit nový firmware.
5.2 Testovací procedury

Testování každé funkce bude prakticky realizováno pomocí testovací procedury. Každé funkci je přiřazena sada testovacích procedur, pomocí kterých jsou testovány všechny možné funkcionality dané funkce.

Testovací procedury jsou implementovány jako spustitelné skripty. Skripty jsou psány v jazyce BASH a pomocí testovacího API. Spouštěnému skriptu se jako parametr předává identifikační číslo testovaného modelu a identifikační číslo testované verze firmware. Skript spouští testy na daném routeru a porovnává výsledky testů spuštěných na routeru s výsledky testů spuštěných na modelu testovaného zařízení.

Praktické připojení na konkrétní zařízení je realizováno pomocí telnet či SSH spojení. Tento princip nám umožňuje testovat jakékoliv zařízení umožňující komunikaci jedním z těchto protokolů. Dále je možné přidání dalších protokolů, ale u navrhované testovací aplikace si vystačíme pouze s těmito protokoly. Připojení na model zařízení je realizováno pomocí databáze, kde jsou uložena data modelu. Pro přístup k modelu i testovanému zařízení slouží řada programů z testovacího API, jenž jsou popsány v samostatné kapitole.

5.3 Model zařízení

Každému výrobku je ve webovém rozhraní vytvořen model. Model obsahuje informace o daném výrobku jako je název testovaného výrobku, číslo portu, do kterého je výrobek zapojen, IP adresa primárního portu výrobku a výchozí protokol komunikace se zařízením a další informace o tomto výrobku. Dále jsou tomuto modelu přiřazeny funkce, které daný model podporuje. Každý testovaný výrobek má tedy k sobě přiřazen model obsahující informace o tomto výrobku a množinu funkcí, jenž daný výrobek podporuje a mají být na něm testovány.

Při testování výrobků testovací program testlab dle parametrů modelu naváže s testovanými výrobky spojení. Po úspěšném navázaní spojení program spouští na daném zařízení testy definované v samotném modelu výrobku. Dále po provedení každého testu program porovnává výsledky testů spuštěných na testovaném zařízení s očekávanými výsledky testů spuštěných na modelu testovaného zařízení. Výsledky testů a případné chybové hlášky program zapisuje do databáze pro možnost vytvoření reportu z každého testování.
5.4 Příklad modelu zařízení


Pokud je model výrobku úspěšně přidán do testovacího zařízení, tak v dalším spuštění testování je nový výrobek již testován. Testovací program se připojí k testovanému zařízení a podle modelu spouští na testovaném zařízení jednotlivé testy a výsledky porovnává s výsledky testů provedených na modelu.


![Obrázek 24 Příklad modelu routeru LR77 v2F RS232](image)

5.5 Výhody modelového přístupu

Díky tomuto přístupu není potřeba při přidávání nového výrobku psát nové testovací skripty pro testování výrobku, ale pouze navolíme vlastnosti výrobku. Výrobek je testován dle testovacích procedur navoněných funkcí. Pokud je vyvinuta nová funkcionalita, jsou pro tuto funkcionalitu napsány testovací skripty a všem zařízením podporujícím tuto funkcionalitu se pouze přidá nová funkce v jejich modelu a není proto potřeba psát testovací procedury pro všechny možné routery zvlášť. Pokud dojde ke změně zapojení
v testovací laboratoři, opět není nutné přepisování všech testovacích skriptů routerů, kterých se tato změna týká, ale pouze je změněn model zařízení a testování funguje dále beze změn.
6 Testovací program

Testovací systém běžící na serveru testovací laboratoře se skládá z několika samostatných částí. Základem celého systému je databáze uchovávající všechny informace o struktuře testovací laboratoře, informace o všech modelech testovaných zařízení a data s výsledky jednotlivých testů. Další součástí je program testlab, jenž se stará o celý průběh testování. Několika málo přepínačů lze nastavit průběh testování. Další součástí je sada programů nazývajících se testovací API, tyto programy slouží k usnadnění psaní jednotlivých testů. Nedílnou součástí testovacího systému jsou testovací skripty, které lze rozdělit na skripty pro stáhnutí projektu, kompilaci prostředí, testování zařízení a úklid projektu. Tyto testovací skripty odpovídají testovacím procedurám v postupu testování založeného na modelech popsaného v předchozí kapitole. Poslední součástí testovacího systému je webové rozhraní pro sledování výsledků testování a nastavování chování testovacího systému.

6.1 Adresářová struktura testovacího systému


Druhá část adresářové struktury testovacího systému obsahuje soubory a adresáře měnící se v průběhu běhu testovacího systému. Tato část se nachází v adresáři /var/testlab a je rozdělena na následující adresáře. Prvním adresářem je clean obsahující skripty pro zajištění úklidu po překladu jednotlivých platformem. Adresář compile obsahuje skripty zajišťující kompilaci jednotlivých výrobků všech platform každého projektu. Pro každý projekt je v tomto adresáři sada skriptů, platforma a produkt se zadává jako parametr. V adresáři checkout naleznete skripty starající se o stáhnutí či aktualizaci zdrojových kódů každého projektu. Pro stahování zdrojových kódů se využívají repozitáře a konkrétně bude využit verzovací systém GIT. Adresář source slouží k uchování zdrojových kódů jednotlivých projektů. Projekt je pracovním adresářem, kam jsou kopírovány zdrojové kódy jednotlivých platform, a kde jsou následně překládány. Dále je tento adresář rozdělen do jednotlivých podadresářů dle jednotlivých verzí překládaného firmware. V každém adresáři releasu jsou adresáře pro každou testovanou platformu. Adresáře jednotlivých verzí se před ukončením kompilace odstraňují, jelikož dál nejsou potřeba a zabírají velký prostor na disku. Přeložené firmware všech výrobků je ukládán do adresáře firmware a do podadresáře s názvem identifikačního čísla verze, pro který byl
firmware přeložen. Tento adresář bude později přesunut na jiný disk vzhledem k omezené kapacitě ssd disku. Testovací skripty se nacházejí v adresáři tests. Adresář tests se dále dělí na podadresáře s názvy funkcí. V adresářích s názvem funkcí se nacházejí jednotlivé testovací skripty, jejichž název je shodný s testovací procedurou. Posledním adresářem této části testovacího systému je adresář logs. V adresáři logs se ukládají logy z jednotlivých fází testování, například logy ze stahování zdrojových kódů, ze samotné kompilace všech výrobků a z úklidu po překladu. Adresář je členěn podle typu logu a dále podle verze testovaného firmware. Adresář s logy bude postupně přesunut na jiný disk obdobně jako adresář firmware. Chybové logy z prováděných testů se ukládají do databáze.

Obrázek 25  Adresářová struktura testovacího systému

Třetí část testovacího systému se nachází v adresáři /var/www/html. Tuto část testovacího systému tvoří webové stránky testovacího systému.

### 6.2 Struktura databáze

Všechny informace o testovaných zařízeních a výsledcích testů jsou uloženy v databází. K těmto účelům byla využita MySQL databáze. K databázi má přístup samotný testovací program testlab, všechny programy testovacího API a v neposlední řadě webová aplikace sloužící k administraci modelů testovaných zařízení a reportování výsledků testů. Pro organizované uchovávání všech dat byla navržena základní struktura databáze, která se časem s přibývající funkcionalitou testovacího systému může rozšiřovat. Jednotlivé tabulky této struktury jsou popsány v samostatných sekcích.

#### 6.2.1 Tabulka fwreleases

Položka idfwerleases je primárním klíčem tabulky, položka date uchovává datum a čas vzniku releasu a položka type určuje typ vydání firmwareu.

Obrázek 26  Tabulka fwerleases

6.2.2 Tabulka platforms

Tabulka platforms obsahuje informace o jednotlivých platformách. Platforma je skupina výrobků postavena na společných zdrojových kódech a na jednom typu procesoru. Platformy se dále dělí na výrobky. Tabulka obsahuje prozatím následující 4 položky. Položka idplatforms, která je primárním klíčem tabulky, položka name slouží k uložení názvu platformy, dále položky timeout_checkout a timeout_build sloužící k nastavení timeoutu skriptu pro stažení zdrojových kódů platformy a pro přeložení zdrojových kódů platformy.

Obrázek 27  Tabulka platforms

6.2.3 Tabulka products


Obrázek 28  Tabulka products
6.2.4 Tabulka checkout


Obrázek 29 Tabulka checkout

6.2.5 Tabulka builds platform

Tabulka builds_platform slouží k ukládání výsledků překladu celé platformy, čili všech výrobků dané platformy. Tabulka obsahuje celkem 4 položky. Položka idbuilds_platform je primárním klíčem tabulky. Položka idplatforms odkazuje na tabulku platforms a udává platformu, které se výsledek překladu týká. Položka idfwreleases odkazuje na tabulku fwreleases a udává k jakému vydání firmwareu je zdrojový kód překládán. Poslední položka state představuje stav ukončení překladu firmwareu celé platformy.

Obrázek 30 Tabulka builds platform

6.2.6 Tabulka build product

6.2.7 Tabulka routers


6.2.8 Tabulka functions

Další tabulkou pomocí níž se tvoří model testovaného zařízení je tabulka functions. Tabulka functions sdružuje data o jednotlivých funkcích, které mohou testované výrobky podporovat. Tabulka obsahuje následující položky, definující informace o dané funkci. Položka idfunctions je primárním klíčem tabulky. Položka name definuje název, pod kterým je daná funkcionalita reprezentována v testovacím systému. Poslední je položka order určující pořadí v jakém mají být funkcionality testovány.
6.2.9 Tabulka dependencies


6.2.10 Tabulka routers has functions

Další tabulkou tvořící model testovaného zařízení je tabulka routers_has_functions. Pomocí této tabulky jsou každému modelu přiřazeny funkce, které je možné testovat. Přiřazení je realizováno pomocí dvou cizích klíčů odkazujících do tabulek routers a functions. Do tabulky routers odkazuje položka idrouters a položka idfunctions odkazuje do tabulky functions.

6.2.11 Tabulka procedures

Tabulka procedures již neslouží k uchování dat z abstraktního pohledu testování zařízení na modelech. Tabulka uchovává jednotlivé spustitelné procedury sloužící

![Obrázek 36 Tabulka procedures](image)

### 6.2.12 Tabulka tests router

Výsledky testování jsou ukládány do tří různých tabulek pro jednodušší reportování výsledků testování. První tabulkou je tests_router do níž jsou ukládány informace o testování celého zařízení. Za úspěšný se tento test považuje, pokud všechny procedury spuštěné na testovaném zařízení byly ukončeny úspěšně. Tabulka obsahuje čtyři následující položky. Položka idtests.router je primárním klíčem tabulky. Položka idroutes odkazuje na tabulku routers a určuje jakému zařízení je výsledek testu určen. Položka idfreleases odkazuje na tabulku freleases a určuje k jakému vydání firmwaru je výsledek testu přiřazen. Poslední položkou je samotný výsledek testu a to položka result.

![Obrázek 37 Tabulka tests router](image)

### 6.2.13 Tabulka tests function

Druhou tabulkou sloužící k ukládání výsledků testů je tabulka tests_function. Tato tabulka sdružuje výsledky všech testovacích procedur dané funkce na jednom testovaném zařízení. Test funkce je považován za úspěšný, jestliže všechny testované procedury této funkce proběhly úspěšně. Tabulka tests_function obsahuje pět následujících položek. Položka idtests_function je primárním klíčem tabulky. Položka idfunctions odkazuje na tabulku functions a definuje testovanou funkci. Položka idfreleases odkazuje na tabulku freleases a definuje k jakému testovanému firmwaru je výsledek testování
6 Testovací program

fungce přiřazen. Položka idrouters odkazuje na tabulku routers a definuje jakému zařízení je výsledek testu přiřazen. Poslední položka result určuje samotný výsledek testu.

Obrázek 38 Tabulka tests function

6.2.14 Tabulka tests procedure


Obrázek 39 Tabulka tests procedure

6.2.15 Tabulka logs


36
6.3 Popis programu


Obrázek 41  Základní schéma testovacího programu

Po rozepírání parametrů se provádějí přípravné kroky pro samotné testování. V prvním kroku je připravena nová verze a následně vložena do databáze. V adresáři projekt je vytvořen nový adresář s názvem identifikačního čísla testovaného releasu. Po těchto krocích je založen nový release firmwaru a může se přejít k samotnému testování.
6.3.1 Checkout

V prvním kroku testování jsou stahovány zdrojové kódy testovaného projektu. Název projektu je předán programu jako parametr a detailní informace o projektu jsou získány z databáze. Po získání všech informací je spuštěn program checkout a v hlavním programu se pouze čeká na ukončení programu checkout.


Po ukončení aktualizace zdrojových kódů projektu jsou zdrojové kódy kopírovány do pracovních adresářů jednotlivých platform. V těchto pracovních adresářích jsou zdrojové kódy dále kompilovány.

6.3.2 Compile

Další fázi každěho testování je kompilace zdrojových kódů všech platform. Pro ukládání přeložených firmwarů je vytvořen adresář s názvem identifikačního čísla verze v adresáři firmware. Po vytvoření tohoto adresáře je pro každou platformu spuštěn program compile. V programu testlab se dále čeká na ukončení všech programů compile.

6.3.3 Remote server

Po ukončení překladu všech výrobků končí fáze kontinuální integrace a začíná systémové testování na konkrétním hardwaru. Pro účely testování se s každým zařízením udržuje permanentní spojení pomocí jednoho z protokolů telnet nebo SSH. Způsob komunikace pomocí permanentního spojení byl zvolen, jelikož díky velmi častému opětovnému připojování zařízení již dále nepřijímala další žádosti o připojení. Nejdříve jsou z databáze vybrány všechny testované zařízení a pro každé testované zařízení je spuštěn program remote server udržující permanentní spojení.


Program remote server po zpracování všech parametrů vytvoří pojmenovanou rouru, pomocí nichž bude komunikovat s programy testovacího API. Pojmenované roury jsou vytvářeny v adresáři /tmp. Název roury, pomocí které server naskoučá příchozím požadavkům je remote_server_id_pipe, kde id je identifikační číslo zařízení. Název roury, pomocí které server předává odpověď dotazovanému programu, je remote_client_pid_pipe, kde pid je process id dotazovaného programu. Po vytvoření komunikační roury přichází fáze, kdy remote server čeká na příjem požadavku. Čekání na příjem požadavku je provedeno blokováním otevřením čtecí pojmenované roury. Zde je program blokován až do otevření této roury klientským programem. Pokud klientský program z testovacího API otevře tuto rouru a zapíše do ní data, remote server je přečte a uloží do datové struktury. Z datové struktury zprávy je zjištěno o jaký druh žádosti se jedná a žádost je provedena. Remote server podporuje následující typy žádostí:

- Vykonání příkazu
- Změna adresy připojení
- Změna portu připojení
- Změna přihlašovacího jména
- Změna přihlašovacího hesla
- Změna protokolu pro přihlášení
- Nastavení defaultní konfigurace
- Restart připojení do routeru
- Ukončení aplikace
- Informace o aktuální adrese
- Informace o aktuálním portu
- Informace o aktuálním uživateli
- Informace o aktuálním hesle
- Informace o aktuálním přihlašovacím protokolu

Po provedení příkazu remote server odpovídá jednou z těchto možných odpovědí:
Obrázek 42 Schéma komunikace s testovaným výrobkem

- Úspěšné vykonání
- Chyba ve skriptu

Funkce všech žádostí jsou zřejmě jasně z jejich názvů a tak dále bude detailně popísána hlavní a nejvíce používaná žádost vykonání příkazu. Samotné vykonání příkazu je provedeno podle zvoleného protokolu.

Telnet

neúspěchu při přihlášení do routeru je ukončeno TCP spojení se zařízením.

Po úspěšném navázání spojení se mohou do routeru posílat příkazy, které by měl router vykonat. Samotné posílání příkazů do routeru je velice jednoduché. Příkaz je poslán pomocí vytvořeného TCP spojení do routeru a následně jsou přijímány data, dokud není přečtena sekvence znaků CR, LF, # a mezeera. Pro zamezení chyb při přijmu z routeru je nastaven timeout 60 s. Po přečtení celé odpovědi z routeru je odpověď zpracována a pomocí pojmenované roury odeslána dotazovanému programu. Ukončení téhnet spojení s testovaným zařízením je prováděno uzavřením daného TCP spojení.

**SSH**

Komunikace s testovaným zařízením pomocí SSH protokolu je řešena pomocí programu Plink. Program Plink je z balíčku Putty-tools a vykonává funkci SSH klienta. Důvodem proč je použit program Plink mimo stanoveného SSH klienta je možnost zadání přihlašovacího hesla jako parametr programu. Inicializace SSH spojení je tedy prováděna spuštěním programu Plink v novém procesu. Komunikace remote serveru s programem Plink probíhá pomocí standardního vstupu a výstupu. Před spuštěním programu Plink je standardní vstup a standardní výstup programu Plink připojen na nepojmenované roury vytvořené v programu remote server pro komunikaci s programem Plink. Po spuštění programu Plink je čten výstup programu Plink a je kontrolováno úspěšné navázání spojení s testovaným zařízením. V případě úspěšného spojení s testovaným zařízením inicializace funkce vrací PID programu Plink. Pokud spojení nebylo úspěšně navázáno, program Plink je ukončen a funkce vrací informaci o neúspěšném připojení.


### 6.3.4 Test

Po úspěšném navázání spojení se všemi testovanými zařízeními je započato samotné testování těchto zařízení. Testování bude rozděleno do několika sekcí podle typu provádění testů. V první fázi bude implementováno testování jednotlivých zařízení. Tento přístup testuje každý router samostatně a testování všech zařízení probíhá paralelně.

V dalších fázích bude implementováno testování zařízení proti sobě, typickým testem může být spojení dvou zařízení IPsec tunelem. Těmito typem testování bude sekvenční testování zařízení, kdy při testování bude využit jedinečný zdroj, tudíž je nutné všechny zařízení testovat postupně.

Pro otestování všech zařízení je pro každé zařízení spuštěn program test se dvěma parametry. Prvním parametrem je identifikátor testovaného rejeasu. Druhým parametrem programu test je identifikační číslo testovaného zařízení. Po kontrolé parametrů, program test z databáze vybere všechny podporované funkce testovaného zařízení. Při zpracovávání každé funkce jsou z databáze vybrány všechny testovací procedury určené pro testování dané funkce. Nyní je vše připraveno pro postupné spuštění všech testovacích procedur.
Před samotným spuštění je nejdříve kontrolovaná existence testovacího skriptu. Dále jsou vytvořeny nepojmenované roury pro předávání informací ze skriptu do testovacího programu. K předávání výsledné hodnoty testu je na rouř testovacího programu připojen standardní výstup spuštěného skriptu. Předávání chybových hlášek z testovaného skriptu je prováděno propojením chybového výstupu a nepojmenované roury testovacího programu. Po správném nastavení vstupů a výstupů je spuštěn testovací skript. Při běhu testovacího skriptu samotný program čeká, jestli mu ze skriptu nepřicházejí nějaká data, či již nevyprší timeout pro provedení skriptu. Úspěšnost provedení testu je vyhodnocována z návratové hodnoty testovacího skriptu. Po ukončení skriptu je do databáze zapsán výsledek testu, hodnota testu předaná přes standardní výstup skriptu a chybové hlášky předané přes chybové výstup skriptu.

6.3.5 Clear

Nyní, když jsou firmwary přeloženy a testy provedeny, přichází fáze úklidu, kdy jsou ukončovány běžící programy a odstraňovány dočasné soubory. Nejdříve jsou ukončovány programy remote server udržující spojení s testovanými zařízeními. Ukončení programu je provedeno spuštěním programu remote exit, jenž pouze odesílá příslušnému remote server procesu příkaz pro ukončení aplikace. Po odeslání žádosti o ukončení všem programům remote server, hlavní program čeká na ukončení všech programů remote server a remote exit. V dalším kroku je provedeno odstranění všech přeložených platform. Na každou platformu je zavolán program clear, který podle projektu spouští příslušný skript clear pro úklid platformy. Úkol skriptu clear je odstranit soubory, pro jejichž smazání by byla potřeba práva root, tudíž by z programu testlab nešly smazat. Například platforma překládána pomocí ltbu volá pouze ltb -m distclean. Průběh tohoto skriptu je obdobný jako u skriptu checkout a compile logován do souborů v adresáři logs. Po vyčištění všech platform jsou všechny soubory v adresáři project odstraněny. Nakonec je uvolněna všechna dynamicky alokovaná paměť, uzavřen log a program testlab je ukončen.
7 Testovací API

Další nedílnou součástí testovacího systému je framework pro zjednodušení psaní jednotlivých testů. Testovací API tvoří sada programů, kde každý z nich vykonává specifický, a v testování často opakující, úkol. Název každého programu se skládá s předpony tl_ a názvu daného programu, který většinou vystihuje jeho účel. Hlavním cílem vytvorění testovacího API je zjednodušení psaní samotných testů, a tím i převedení psaní testů z programátorů k testerům.

V první fázi je k dispozici základní sada programů testovacího API, která je níže popsána. Dále bude podporováno dopisování vlastních programů testovacího API. Podporované programovací jazyky, pro psaní vlastních API programů, jsou jazyky C a Bash. Pro jazyk C je možné využít knihovny pro komunikaci s remote serverem a komunikaci s databází testovacího systému. Při psaní nových uživatelských programů testovacího API v Bashu je možné využít pro komunikaci s remote serverem program tl_remote.

7.1 Knihovna cspipe

Knihovna libtl_cspipe.so slouží ke komunikaci s testovaným zařízením prostřednictvím remote serveru. Knihovna definuje názvy pojmenovaných rout, prostřednictvím kterých je komunikováno s remote serverem, dále velikost přijímačeho a odesílacího bufferu a výčet typů všech příkazů, které remote server podporuje. Pro výměnu dat přes roury slouží struktura message_remote, která obsahuje proces ID dotazujičeho programu, položku z výčtu podporovaných příkazů a buffer s posílanými či přečtenými daty.

Knihovna obsahuje čtyři funkce pro komunikaci s remote serverem.Inicializace spojení s remote serverem je prováděna pomocí funkce client_starting. Funkce se jako parametr předává proces ID běžícího programu a při úspěšném připojení funkce vrací file deskriptor otevřeného spojení. Spojení je možné ukončit pomocí funkce client_ending, které jsou předány parametry file deskriptor otevřeného spojení a process ID běžícího programu. Knihovna dále obsahuje funkce pro zápis do otevřené serverové roury a čtení z klientské roury. Všechny tyto operace je možné provést funkci pipe_request, které se jako parametr předává typ žádost, buffer s obsahem předávané zprávy a buffer pro nahlášení přijaté zprávy. Tato funkce vrací -1 při neúspěchu při spojení s remote serverem nebo testovaným zařízením a jakékoliv kladné číslo při správné odezvě remote serveru.

7.2 Knihovna database

7.3 Knihovna utils

Poslední vlastní používanou knihovnou je knihovna utils. Knihovna utils obsahuje funkce pro změny v systému. První funkce close_all_fds slouží k uzavření všech otevřených souborových deskriptorů. Druhá a zatím poslední funkce této knihovny zajistuje rekurzivní mazání adresářů.

7.4 Program checkproduct

První program checkproduct slouží k získání názvu firmwaru patřícího do daného výrobku. Syntaxe tohoto příkazu je `tl_checkproduct <id>`. Jediným parametrem id zadáváme identifikační číslo zařízení. Program zjistí název firmwaru z databáze pomocí databázového dotazu z knihovny database.

7.5 Program remote

Základním programem pro komunikaci s testovaným zařízením pomocí remote server je program remote. Syntaxe programu remote je `tl_remote <id> <command>`. První parametr id určuje identifikační číslo zařízení, s kterým chceme komunikovat. Druhým parametrem command je samotný příkaz, jenž bude v testovaném zařízení proveden. V případě úspěšného navázání spojení s testovaným zařízením vypíše program výstup provedeného příkazu na testovaném zařízení a ukončí se s identickým návratovým kódem. Jestliže se z jakéhokoliv důvodu nepodařilo k testovanému zařízení připojit, tak je na chybový výstup vypsána příslušná hláška a program se ukončí s návratovým kódem 120. Program remote pro komunikaci s remote serverem používá knihovnu cspipe.

7.6 Program remotechange


7.7 Program remoteinfo

Program remoteinfo slouží k získání aktuálních informací o parametrech připojení s testovaným zařízením. Syntaxe programu je `tl_remoteinfo [-p] or [-i] or [-u] or [-s] or [-t] <id>`. Program zpracuje pouze jeden zadaný přepínač, a to ten poslední zadaný. Zadáním přepínače -p získáme port, pomocí kterého je testované zařízení připojeno. Při zadání přepínače -i program vrátí IP adresu zařízení, ke kterému je remote server připojen. Přepínače -u a -s slouží ke zjištění uživatelského jména a hesla, které byly použity k připojení do testovaného zařízení. Při zadání posledního
možného přepínáče -t dostaneme zpět protokol pomocí, kterého jsme do testovaného zařízení připojeni. Parametrem id zadáme identifikátor dotazovaného zařízení. Program pouze odešle žádost o informaci jednoho z parametrů remote serveru a vytiskne příchozí odpověď na standardní výstup.

7.8 Program routerready


7.9 Program status

Programem status je možné z testovaného zařízení vyčíst status zařízení, či přímo pouze hodnotu položky statusu zařízení. Syntaxe tohoto programu je `tl_status <id> <category> [subcategory]`. Prvním parametrem id je zadán identifikační číslo zařízení, ze kterého má být status vyčten. Druhým parametrem category určíme požadovanou kategorii statusu, například lan pro informace o sítovém rozhraní. Poslední volitelnou položkou subcategory určíme jakou položku v dané kategorii má program vytištíknout. U již zmíněné kategorie lan si můžeme nechat vrátit například položku MAC Address.

7.10 Program updateconf


7.11 Program updatefw

Nahrávání nového firmware do testovaného zařízení je prováděno pomocí programu updatefw. Syntaxe programu updatefw je `tl_updatefw -r <release> -f <firmware> [-d <fwdir>] <id>`. Prvním parametrem release určíme z jakého přeloženého releasu

7.12 Program slog


7.13 Program rlog


7.14 Program klog


7.15 Program changeparam

Program changeparam slouží k provedení změny jednotlivých položek konfigurace testovaného zařízení. Změnu se provádí příkazem changeparam, není nutné žádné nahrávání konfigurace do zařízení. Syntaxe programu remotedebug je tl_changeparam -f <function> -p <param> [-r <profile>] <id> <value>. Prvním parametrem function je určena funkce, u které se parametr mění, například ppp u mobilního spojení. Druhým

46

### 7.16 Program mobilready

Program mobilready slouží k čekání na navázání mobilního spojení testovaného zařízení. Syntaxe programu je \texttt{tl_mobilready [-t \{timeout\}] \{id\}}. Prvním volitelným parametrem timeout je možné změnit maximální čas čekání na připojení zařízení do mobilní sítě. Defaultní čas čekání je 120 s. Druhým parametrem id je určeno testované zařízení. Program provádí kontrolu čekání na mobilní spojení kontrolou přiřazením IP adresy mobilnímu rozhraní. Kontrola IP adresy je prováděna každou sekundu až do vypršení timeoutu nebo do úspěšného navázání spojení.
8 Uživatelský interface

Komunikaci testlabu s jeho uživateli obsahuje webová aplikace. Webové aplikace je vytvořena pomocí technologií html, css, php a javascript. Pomocí webové aplikace bude možné prohlížet modely všech zařízení a jednotlivé modely upravovat. Model zařízení obsahuje všechny informace o testovaném zařízení, například IP adresu zařízení, informace o vložené SIM kartě a podporované funkce. V neposlední řadě je možné zjistit informace o všech vykonaných testech zařízení a překladech firmwarů. Jednotlivé možnosti zobrazení a úpravy informací jsou popásány v následujících sekcích.

8.1 Sekce build

Sekce build zobrazuje všechny informace o stahování zdrojových kódů z repozitářů a jejich následné kompilaci. Sekce je prozatím rozdělena na tři stránky list release, build platforms a build products. První stránka zobrazuje informace o jednotlivých releasech a ve zbylých dvou stránkách naleznete informace o překladech všech firmwarů.

8.1.1 Stránka List releases

První stránka list releases sekce build informuje o všech vytvořených verzích. Release je vytvořen při každém spuštění programu testlab a jsou k němu vázány všechny informace celého průběhu testování. Stránka obsahuje pouze jednu tabulku, kde každý řádek odpovídá jednomu vytvořenému releasu.

<table>
<thead>
<tr>
<th>TestLab Goliath</th>
<th>Build</th>
<th>Test</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Builds [Routers]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>List releases</td>
<td>Build platforms</td>
<td>Build products</td>
<td></td>
</tr>
</tbody>
</table>

**Obrázek 43** Stránka release sekce build

Tabulka má celkem osm sloupců. První sloupec state informuje o výsledku stažení nebo aktualizaci zdrojových kódů testovaného projektu. Sloupec state rozlišuje dva stavy jimiž jsou modrá kulička při úspěšném stažení zdrojových kódů a červená kulička při neúspěšném stažení zdrojových kódů. Sloupec Build ID zobrazuje jedinečné číslo identifikační každý release, které musí být v jednom projektu u každého releasu unikátní. Sloupec build version zobrazuje verzi překládaných zdrojových kódů. Build date zobrazuje datum a čas, kdy bylo testování spuštěno. Sloupec build type informuje, zdali testovaný release byl ve fázi BETA nebo ve fázi ostrého releasu určeného pro zákazníky. Sloupec repository branch informuje, z jaké větve repositáře byl firmware
překládán a sloupec repository hash zobrazuje hash posledního commitu testovaného zdrojového kódu. V posledním sloupci je umístěn odkaz na log z průběhu stahování či aktualizace zdrojových kódů testovaného releasu.

8.1.2 Stránka Build platforms


![Obrázek 44 Stránka platform sekce build]

8.1.3 Stránka Build products

V případě úspěšného přeložení firmwareu jsou ve sloupci firmware odkazy na soubory s přeloženým firmwarem. Poslední sloupec logs obsahuje odkazy na logy z každého překladu, pomocí nichž lze jednoduše hledat případné chyby při překladu. Druhá tabulka má totožný obsah i význam jako u předchozí stránky build platforms.

![Obrázek 45 Stránka product sekce build]
8.2 Sekce test


8.2.1 Stránka Devices


Obrázek 46 Stránka product sekce build

8.2.2 Stránka Functions


Obrázek 47 Stránka product sekce build
8.2.3 Stránka Procedures

Stránka procedures zobrazuje výsledky všech spuštěných testovacích procedur nebo pouze výsledky testovacích procedur jedné vybrané funkce. Funkci si lze vybrat na předchozí stránce functions. Rozložení a funkce stránky jsou zcela totožné se stránkou functions, pouze místo funkcí jsou zobrazovány již konkrétní testovací procedury a jejich výsledky.

Obrázek 48 Stránka product sekce build

8.2.4 Stránka Logs


Obrázek 49 Stránka product sekce build
9 Návrh testů pro Conel routery

V této fázi je připraven celý systém pro testování routerů. Testovací laboratoř se všemi výrobky je kompletně postavena. Testovací server je nainstalován a nakonfigurován. Mám navrhnout databázi a adresářovou strukturu testovacího systému. Systém pro stahování zdrojový kódů, kompilování projektů a následné spouštění testů již také funguje správně. Pro provádění jednotlivých kroků testů jsou k dispozici programy z testovacího API a výsledky všech testů je možné zobrazit pomocí webových stránek.


Pro základní testování funkčnosti celého portfolia výrobků společnosti Conel byly navrženy následující funkce a testovací procedury. V první fázi implementace testovacího systému budou spouštěny tyto nízké popisované procedury.

Testovací skripty jsou spouštěny se dvěma parametry. Prvním parametrem je identifikační číslo testovaného zařízení. Pomocí tohoto čísla je možné komunikovat s daným zařízením a zjišťovat detailní informace o zařízení z databáze pomocí programů popsaných v sekci věnující se testovacímu API. Druhým parametrem je předáváno identifikační číslo testovaného releasu.

9.1 Checkout


9.2 Compile

Pro každý projekt je dále napsán skript zajišťující kompilaci daného projektu a vybraného produktu. Skript je pouštěn se dvěma parametry. Prvním parametrem je cílový adresář kam má být výsledný firmware nakopírován a druhým parametrem je název překládaného výrobku. Prozatím se pro všechny testované projekty používá buildovací systém lttb, tudíž skripty zajišťující překlad projektů vypadají velice obdobně. Každý
skript je prováděn v adresáři, kde je projekt stažen. V prvním kroku je otevřen adresa-
sář ltb. Dále pokud se překládá první výrobek platformy, tak je vybrána zpracovávaná
platforma. Samotné vybrání platformy je provedeno ve dvou krocích. V prvním kroku je
spuštěn skript platform s parametrem název platformy. Skript platform byl do buildova-
cího systému doplněn v rámci testovací laboratoře, jelikož ltb nepodporoval překládání
bez interakce uživatele. Výběr platformy je dokončen spuštěním programu ltb v konfi-
guračním režimu. V dalším kroku je vybrán produkt, který má být překládán. Vybrání
produktu je provedeno zapsáním jeho názvu do souboru appname. Nyní jsou smazány
přeložené části závisící na výrobku a spuštěn samotný překlad. V případě úspěšného
ukončení překladu jsou přeložené firmwarey nakopírovány do adresáře předaného jako
parametr skriptu. Překládání projektů pomocí skriptů bylo zvoleno z důvodu univer-
zálosti a jednoduchém přidávání nových projektů založených na různých buildovacích
systémech.

9.3 Clean

Před ukončením testování je prováděn úklid po kompilaci projektů. Aby bylo možné
smazat všechny soubory a adresáře zdrojových kódů a soubory vytvořené při samotném
překladu je nutné provést clean nad projektem. Jelikož clean každého projektu je pro-
váděn odlišným způsobem, je pro každý projekt vytvořen vlastní skript. Pro projekty,
které jsou zatím testovány v testovací laboratoři je nejúčetnější otevřít adresář ltb. Samotný
úklid po překladu je proveden spuštěním programu ltb v módě distclean. Na závěr se
skript vrátí do výchozího adresáře a je ukončen s návratovým kódem programu ltb.

9.4 Funkce firmware

Základní funkcionalitou každého výrobku je možnost nahrání nového firmwareu. Na funkci
nahrání nového firmwareu je založeno veškeré další testování. Pokud se nepodaří nahrát
nový firmware není důvod testovat jakoukoliv funkcionalitu, jelikož tyto testy neodpo-
vídají danému firmwareu. [7] [8]

9.4.1 Procedura upload

První testovací procedura funkce firmware testuje nahrání firmwareu do testovaného
zařízení. V prvním kroku je zjištěn název firmwareu testovaného zařízení z databáze
testovacího systému. Dále je správný firmware nahrán do testovaného zařízení pomocí
utility updatéfw z testovacího API. Test je ukončen úspěšně, pokud se podaří nahrát
firmware do zařízení.

9.4.2 Procedura start

Další testovací procedura funkce firmware testuje, jestli router po nahrání firmwareu
nastartoval. V tomto testu se nejdříve jednu sekundu čeká na reboot routeru po ukončení
programování, jelikož pokud by se router testoval ihned po nahrání, testoval by se ještě
běžící stary firmware. Dále je spuštěn program testující připojení routeru do testovací
sítě. V tomto programu je nejdříve testování ping na router a v případě úspěšného pingu
je testováno odezva routeru na příkaz echo. Test končí úspěšně, pokud se na routeru
podarí ping, a také se k němu podaří připojit jedním z podporovaných protokolů. Tento
test vrací také čas startupu routeru v sekundách.
9.4.3 Procedura check

Poslední procedura funkce firmware check testuje shodu verzí firmware v routeru s nahránou verzí firmware testovacím systémem. V první fázi je z databáze zjištěn název firmware testovaného výrobku. Pomočí názvu firmware je nalezen soubor s verzí firmware dodávaný s přeloženým firmware a jeho obsah uložen do proměnné. Po zjištění požadovaného firmware je zjištěn aktuální firmware v zařízení pomocí programu status. Nakonec jsou tyto dvě verze porovnány a skript je úspěšně ukončen, pokud jsou obě verze totožné.

9.5 Funkce configuration


9.6 Funkce connect


9.6.1 Procedura telnet

První procedura telnet testuje připojení do routeru pomocí protokolu telnet. Nejdříve je zjištěn původní komunikační protokol, aby bylo možné před ukončením testu protokol zpět obnovit. Komunikační protokol je zjištěn z remote serveru pomocí programu remoteinfo z testovacího API. Dále je remote serveru odeslán požadavek na změnu komunikačního protokolu pomocí programu remotechange. Po změně protokolu je kontrolováno, jestli se protokol oprávdu změnil, a to opět pomocí programu remoteinfo. Na závěr je provedena zkouška komunikace pomocí telnet protokolu. Zkouška komunikace je provedena pomocí programu remote a příkazu echo, následná kontrola komunikace je provedena pomocí protokolu remote serveru.

9.7 Funkce connect ssh

Druhým možným způsobem komunikace s testovanými zařízeními je pomocí zabezpečeného šifrovaného protokolu ssh. S podporou protokolu ssh souvisí i další šifrované typy připojení klasických protokolů. Například zabezpečená varianta ftps protokolu pro přenos dat ftp, dále šifrovaná varianta https webového protokolu http. Zabezpečené
šifrované protokoly nepodporují kvůli malému výkonu starší modely testovaných zařízení, naopak nejnovější modely zařízení podporují pouze tento druh spojení.

9.7.1 Procedura ssh

Procedura ssh testuje spojení s testovaným zařízením přes šifrovaný protokol ssh. Připojení protokolem ssh je prováděno pomocí programu plink, jelikož standardní ssh klient nedovoluje zadání hesla připojení jako parametr. Tato funkcionalita je stejně jako telnet zaintegrována do remote serveru, tudíž je možné se k ssh spojení chovat stejně jako k telnet spojení. Díky tomuto faktu vypadá testovací procedura totožně jako testovací procedura pro telnet s jediným rozdilem, při změně protokolu pomocí programu remotechange se jako parametr nepředává telnet nýbrž řetězec ssh. Podmínka úspěšnosti provedení testu je opět totožná s procedurou telnet.

9.8 Funkce mobile

Nyní již jsou popsány všechny funkcionality nutné k testování všech zařízení a přejdeme k popisu všech ostatních funkcionalit. První testovaná funkcionalita je funkce mobile. Funkce mobile v sobě obsahuje společný základ všech bezdrátových možností připojení. Základními vlastnostmi připojení jsou například přířazení IP adresy, zkouška komunikace či změna parametru určující velikost odcizkového packetu MTU. Jednotlivé vlastnosti této funkcionality jsou popsány v testovacích procedurách testujících tuto vlastnost. Tato funkce naopak nezohledňuje typ spojení bezdrátového modulu se zařízením, či podporované bezdrátové technologie. [9]

9.8.1 Procedura connect

Procedura connect popisuje vlastnost routera připojení pomocí mobilního spojení. Úspěšné připojení do mobilní sítě je definováno přířazením IP adresy rozhraní mobilního spojení. Samotný test probíhá pouze spuštěním programu mobilready, který čeká tři minuty na přířazení IP adresy rozhraní mobilního spojení. Program, tedy i skript, vypisuje pouze čas v sekundách, který čekal na připojení routera do mobilní sítě. Skript je ukončen úspěšně, pokud se routeru podaří připojit do mobilní sítě do tří minut.

9.8.2 Procedura ping

9.8.3 Procedura apn


9.8.4 Procedura address


9.8.5 Procedura operator


9.8.6 Procedura mtu

Procedura mtu popisuje vlastnost mobilního spojení ovlivňující velikost odeslaného packetu. Při změně tohoto parametru bychom měli sledovat změnu velikosti odeslané-

9.9 Funkce mobile edge

Funkcionalita mobile edge popisuje možnost připojení testovaného zařízení do mobilní sítě přes technologie GPRS a EDGE. Tyto technologie nepodporují všechna zařízení, proto byla pro tuto vlastnost vytvořena samostatná funkciální. Testovaná zařízení se k technologii EDGE připojují, pokud v místě použití není dostupná lepší technologie, či zařízení lepší technologií nepodporuje nebo je možné správným parametrem vynutit připojení routeru do mobilní sítě přes tuto technologii.

9.9.1 Procedura type edge

Procedura type EDGE popisuje možnost připojení zařízení do mobilní sítě přes technologie EDGE. Možnost připojení technologii EDGE je testována vyučením technologie EDGE a čekáním na navázání spojení pomocí teto technologie. Testovací tuto vlastnost probíhá následovně. V prvním kroku je zálohován původní typ sítě pro pozdější obnovení a nastavíme nový typ sítě na typ GPRS/EDGE. Z důvodu projevení nastavených změn je restartováno mobilní spojení a dále je čekáno na nové sestavení spojení. Po úspěšném sestavení spojení je pomocí programu status zjištěna technologie, pomocí které je testované zařízení připojeno do mobilní sítě. Test je ukončen úspěšně, pokud zjištěný typ sítě odpovídá nastavené technologii EDGE. Na závěr testu se pouze nastaví zpět původní typ technologie.

9.10 Funkce mobile umts

Funkcionalita mobile UMTS popisuje vlastnost připojení zařízení do mobilní sítě pomocí technologie UMTS a HSPA+. Tyto technologie nepodporují všechna zařízení umožňující mobilní spojení. Například CDMA routery nepodporují GPRS přenosy, EDGE routery podporují maximálně technologie typu EDGE. Naopak LTE routery podporují i nižší technologie UMTS a EDGE. Zařízení komunikuje technologií UMTS, jestliže nepodporuje nebo v místě použití není k dispozici lepší technologie nebo jestliže je správným parametrem vyučena technologie UMTS.

9.10.1 Procedura type umts

Procedura type umts popisuje možnost připojení zařízení do mobilní sítě přes technologii UMTS. Testování technologie UMTS je prováděna totožným způsobem jako testování předešlé technologie EDGE. Jediným rozdílem v samotném testu je, že míst
technologie EDGE se nastavuje technologie UMTS a zjištěná technologie je porovnávána s technologií UMTS.

9.11 Funkce mobile LTE

Funkcionalita mobile LTE popisuje vlastnost připojení zařízení do mobilní sítě pomocí technologie LTE. Technologie LTE je v nynější době nejrychlejší a nejvyspělejší bezdrátovou technologií podporovanou zařízeními společnosti Conel. Tyto routery mají možnost komunikovat mimo LTE technologie i pomocí EDGE a UMTS technologií.

9.11.1 Procedura type LTE

Procedura typu LTE popisuje možnost připojení zařízení do mobilní sítě přes technologie LTE. Testování technologie LTE je prováděno totožným způsobem jako u předchozích technologií EDGE a UMTS. Jediným rozdílem v testovací procedurě je nastavování technologie LTE v konfiguraci a porovnávání vyčtené technologie s technologií LTE.

9.12 Funkce mobile ppp

Funkcionalita mobile ppp popisuje připojení zařízení do mobilního spojení pomocí protokolu ppp. Protokol ppp, neboli point to point protokol, je standardní protokol linkové vrstvy používaný v telekomunikaci. Chování testovaných zařízení, komunikujících tímto protokolem, můžeme změnit například změnou autentizace, či vytaženého čísla. Protokol ppp používají všechna zařízení se staršími modely bezdrátových modulů, protože se jedná o velmi starý protokol, který je v dnešní době nahrazován novějšími a rychlejšími protokoly, například protokolem QMI.

9.12.1 Procedura chap

Procedura chap popisuje vlastnost připojení protokolem ppp pomocí autentizace CHAP. Ve výchozím stavu router vybírá jednu z autentizací PAP či CHAP a pomocí té se přihlásí do mobilní sítě. Výchozí chování lze ověřit změnou parametru autentizace na autentizaci CHAP, poté se bude router přihlašovat do sítě jedině autentizací CHAP. Testování zvolené autentizace CHAP probíhá následujícím způsobem. Nejdříve je založován původní typ autentizace a autentizace je nastavena na CHAP. Pro projevení změny typu autentizace je potřeba restart PPP spojení. Dále je čekáno na nové sestavení spojení se zvolenou autentizací. Po sestavení spojení je v logu hledána hláška od daemona pppd hlásící úspěšné přihlášení do sítě pomocí autentizace CHAP. Test končí úspěšně, jestliže je hláška "CHAP authentication succeeded" nalezena, v opačném případě skript končí s chybovým návratovým kódem.

9.12.2 Procedura pap

Procedura PAP popisuje vlastnost připojení routeru do mobilní sítě protokolem PPP pomocí autentizace PAP. Chování a nastavování autentizace již bylo popsáno v proceduře věnující se autentizaci CHAP. Samotné testování této autentizace je také shodné s testováním autentizace CHAP, až na dvě drobné změny. Při nastavování autentizace je nastavena autentizace PAP a hledaná hláška logu je změněna na "PAP authentication succeeded".

58
9.12.3 Procedura number

10 Testovací laboratoř v praxi


Obrázek 50 Neosazený stojan

Pro první fázi nasazení testování budu muset rozvěst napájení a LAN síť. Napájení jsem řešil pomocí jednoho spínaného zdroje s možností montování na DIN lištu. Samotný zdroj má pouze dvě svorky na výstupní napájení, tudíž jsem vedle zdroje umístil svorkovnice na DIN lištu pro rozvedení napájení všech třiceti testovaných zařízení. Napájecí kabely testovaných zařízení jsou dále vedeny ranžirovacím panelem a rovnoměrně po celé délce stojanu vyvedeny ven. Ethernetové kabely jsem vedl od swiče umístěného ve střední polici stojanu do svrchního ranžirovacího panelu, kde jsou Ethernetové kabely také rovnoměrně vyvedeny ven. Tímto rozložením jsem vytvořil univerzální stojan, kde pro připojení nového zařízení stačí pouze zařízení připevnit na DIN lištu, připojit napájení a Ethernet pomocí volně vísičích kabelů.
Posledním krokem potřebným pro uvedení testovací laboratoře do provozu je připojení všech testovaných zařízení. Do testovací laboratoře jsem zapojil všechny dostupné routery společnosti Conel.

Po zapojení všech testovaných výrobků jsem testovací laboratoř spustil a zahájil testovací provoz, kdy je každý den proveden překlad a test všech výrobků. Při pravidelném testování jsem prozatím nenašel závažné nedostatky. Občasné nepřipojení zařízení do mobilní sítě je způsobeno velmi slabým signálem v dočasném místě umístění testovacího stojanu. Tento problém by měl být vyřešen přemístěním stojanu po jeho kompletním osazení.
Obrázek 52  Dokončené testovací pracoviště

Obrázek 53  Dokončené testovací pracoviště
11 Návrhy na budoucí rozšíření

Testovací laboratoř je v nynějším stavu již plně funkční a pravidelně testuje základní funkčnost všech instalovaných výrobků. Pomocí testovacího API lze snadno doplňovat nové testy testující další funkčnost. Psaní nových testovacích procedur jsem předvedl i testerům a ti již zkouší psát nové testovací procedury. Nová zařízení lze jednoduše přidávat nastavením jejich základních vlastností a přiřazením podporovaných funkcionalit, čili vytvořením jejich modelů.

Pro zvětšení objemu testovaných funkcí, a tím i zlepšení kvality samotného testování, by bylo vhodné testovací systém dále rozvíjet ve směru psaní nových testů. Nové testy by měly testovat všechny jednotlivé funkce routerů. Při dopisování nových testů občas může vzniknout požadavek na nový program testovacího API, avšak tyto požadavky by měly postupem času vymizet. Největším zásahem do testovacího API, který bude muset být v budoucnu udělen je vytvoření sady programů pro automatickou konfiguraci switchů. Ve směru dopisování nových testů nebude vývoj testovací laboratoře nikdy ukončen, jelikož se testované výrobky neustále vyvíjejí, tudíž testovací procedury budou muset být neustále dopisovány.

Dalšími budoucími rozšířeními bude testování všech dostupných rozhraní routerů. Nyní jsou testovány pouze všechny Ethernet rozhraní. V dalších fázích bude přidána testování všech sériových rozhraní zapojením zařízení do testovacích měřičů. Pro účely testování vstupů a výstupů bude potřeba vyvinout přípravky, kde na jedné straně bude komunikační rozhraní pro komunikaci s testovacím serverem a na druhé straně bude řada nastavitelných binárních vstupů a výstupů. Vstupy a výstupy tohoto přípravku budou propojeny se vstupy a výstupy všech testovaných zařízeních.

Správou spoluprací hardwaru se softwarem je možné kontrolovat například měřením spotřeby každého zařízení po nahrání nového firmwareu. Tato hodnota může být také použita k určení průměrných spotřeb nových výrobků. Pro měření spotřeby bude do testovací laboratoře umístěn měřič s jakýmkoliv komunikačním rozhraním pro připojení do testovacího serveru, nejlépe s Ethernet rozhraním. Dále bude navrhována speciální deska, jejíž vstupem bude napájecí napětí pro všechny zařízení testovací laboratoře a komunikační rozhraní pro komunikaci s testovacím serverem. Výstupem desky budou napájecí napětí pro všechny testované zařízení. Měřicí modul bude schopen postupně pomocí relé přívádět jednotlivá napájecí napětí přes ampérmetr a tím měřit spotřebu daného zařízení bez jejich vypnutí.
12 Závěr

Cílem této práce bylo navrhnout, implementovat a ověřit funkčnost metodiky pro automatizované testování s využitím postupů testování podle modelů. Nejdříve byly rozebrány všechny dostupné pohledy na testování. Ze všech metodik byly vybrány způsoby testování implementované v testovacím systémen. Implementována bylo integrační a systémové testování všech vybraných výrobků, ostatní úrovni testování by neprůměrně takovým přísnos konkrétních testovaných výrobků. Další použitá metodika testování vychází ze samotného zadání práce a je to testování podle modelů. Každému zařízení v testovací laboratoři je vytvořen model skladající se základních informací o daném zařízení a všech jeho podporovaných funkcích. Podle těchto informací se na všech zařízeních pouštějí a vyhodnocují testy.

V další fázi byly zkoumány dostupné nástroje pro různé typy testování. Zde nebyl nálezen žádný nástroj, který by byl schopen plně testovat zařízení, pro které má být testovací systém postaven. I úpravy jakéhokoliv ze zkoumaných řešení by byly časově srovnatelné s vývojem nového vlastního systému pro systémové a integracní testování. Ve zbytku práce se tedy počítá s vývojem nového systému a použitím samostatných utilit v rámci modulů nebo API testovacího systému.


Spouštění stahování zdrojových kódů, překladu, jednotlivých testů a dalších pořád v přístupů a testování obsadává hlavní program testlab. Všechny tyto kroky se program testlab snaží provádět co nejvíce paralelně, aby samotné testování bylo co nejrychlejší. Jenom při samotném překladu je při kompilaci program třikrát rychlejší než sekvenční skript. Program testlab vytváří a následně odstraňuje pomocné adresáře potřebné při překladu, dále obstarává vkládání výsledků testů, či chybových hlášek do databáze.

Prozatím bylo napsáno celkem 22 programů testovacího API a knihovna pro jednoduchou komunikaci s testovanými zařízeními a databází testovacího serveru, pomocí níž se dají velmi smadno dopisovat další programy testovacího API. Hotové programy pokrývají základní oblast testování síťových prvků, ale v budoucnu bude určitě potřeba dopisovat nové specializované programy.

Pomocí webového rozhraní je možné zjistit všechny informace o průběhu testování jednotlivých kroků, případně nalézt místo selhání neúspěšného testu či překladu. Dále
je možné prohlédnout si vlastnosti modelů testovaných zařízení, či samotné modely upravovat. Webové rozhraní obsahuje základní potřebnou funkciionalitu. Další funkční, nebo grafické prvky budou doplňovány dle potřeby v průběhu používání testovacího systému.

Nedišnou součástí testovacího systému popisující vlastnosti jednotlivých zařízení jsou testovací procedury. Jak již bylo zmíněno, modely každého zařízení se skládají z jednotlivých funkcí a každá funkce je popsána sadou testovacích procedur. Nyní jsou napsány základní testovací procedury pro každé zařízení, a dále bude vylepšován model každého zařízení dopisováním jednotlivých testovacích procedur. Testovací procedury jsou psány ve skriptovacím jazyku, jelikož tato část testovací laboratoře bude neustále ve vývoji s přicházejícími novými funkcionality testovaných zařízení.

Po navržení teoretické přístupu k testování a následné implementace programů obstávající všechy kroky, byla sestavena a testována kompletní testovací laboratoř. Během testů se neprojevily žádné závažné chyby. Drobné chyby se objevily v implementacích jednotlivých testů. Tyto chyby byly předpokládány a byly již odstraněny. Chyby v testech odráží skutečnost z teoretického úvodu, jenž uvádí, že pokud test je proveden chybně, tak je většinou případů chybný model zařízení nebo je chyba v samotném zařízení.

Cíl práce se podařilo splnit v celém rozsahu, podařilo se navrhnout, implementovat a otestovat metodu automatizovaného testování síťových prvků. Implementace nyní běží na skutečných výrobcích společností Conel. Již nyní testovací systém dokáže otestovat během půl hodiny vše, co by manuálně tester testoval dva pracovní dny. Testovací systém lze díky navrhnutému konceptu snadno rozvíjet ve směru doplňování nových testovaných výrobků i doplňování nových testovaných funkcionalit.
Literatura


