České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí 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:

[1] Isermann, R.: Fault-Diagnosis Systems , Springer 2006, ISBN 3-540-24112-4
[2] Utting, M., Legeard, B.: Practical Model-Based Testing: A Tools Approach, Morgan-Kaufmann 2007, ISBN 978-0-12-372501-1
[3] Shafique, M., Labiche, Y.: A Systematic Review of Model Based Testing Tool Support, Carleton University, Technical Report, May 2010
(http://squall.sce.carleton.ca/pubs/tech_report/TR_SCE-10-04.pdf)

Vedoucí: doc. Ing. Jiří Novák, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

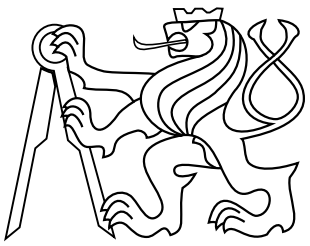
prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 3. 2015

diplomová práce

Metody automatizace testů síťových prvků

Bc. David Felgr



Květen 2015

Ing. Pavel Kopecký, doc. Ing. Jiří Novák, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra Řídící techniky

Poděkování

Chtěl bych poděkovat svému vedoucímu diplomové práce doc. Ing. Jiřímu Novákovi, Ph.D. za prvotní nasměrování při řešení problému. Dále bych chtěl poděkovat kolegům ve vývojovém oddělení za cenné rady při realizaci testovacího systému i při jazykové korekci práce.

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

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 telekomunakční 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 klíču
AT	Protokol komunikace s mobilnímy moduly
GIT	Systém správy verzí

1 Úvod

Tématem této diplomové práce jsou metody automatizace testů síťových prvků. Nejdříve popíši definici testování a různé způsoby, jak je možné testování provádět. Definice testování je podle IEEE Software Engineering Body of Knowledge (SWEBOK 2004) následující:

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

Na způsoby testování lze pohlížet několika pohledy. Prvním pohledem na testování je jeho rozdělení na šest jednotlivých úrovní. Mezi tyto úrovně testování patří testování programátorem, testování jednotlivých jednotek kódu, funkční testování, integrační testování, systémové testování a akceptační testy. Všechny úrovně budou detailně popsány v kapitole používané metody testování. Dalším pohledem na kategorizaci testování je způsob provádění testů. Prvním způsobem je manuální testování, kdy tester manuálně provádí testy podle předem daných testovacích procedur. Dokonalejším způsobem testování je takzvané automatizované testování, kdy testy jsou prováděny automaticky dle předem napsaných testovacích procedur. Testovací procedury jsou psány testery či vývojáři. Všechny testovací procedury musí být přepisovány při změně funkcionality výrobku nebo při přidání nového výrobku. Tímto přístupem je ušetřeno spoustu času, který by byl plýtván opakovaným manuálním testováním identických věcí. Posledním známým a v dnešní době moderním způsobem testování je testování založené na modelech. U tohoto způsobu testování je vytvořen model testované oblasti, či zařízení, a automaticky se generují testovací procedury. Způsob testování založeného na modelech ušetří další čas strávený úpravami a tvorbou dalších testovacích procedur při změně funkcionality softwaru či přidání nového produktu, jelikož je upravován pouze model zařízení. Testovací systém, který bude výstupem této diplomové práce, by měl testovat zařízení na úrovni systémového testování. Systémové testy by měl testovací systém pro konkrétní zařízení vybírat a provádět automatizovaně pomocí testování založeného na modelech.

1.1 Praktické využití

Praktická implementace testovacího systému bude provedena pro testování routerů společnosti Conel. Zadání diplomové práce jsem si vybral, jelikož ve společností pracuji na různých pozicích již 4 roky. Způsoby testování výrobku se během fungování firmy měnily následujícím způsobem. Zpočátku, kdy počet modelů routerů byl velmi malý a routery vyvíjel pouze jeden programátor, bylo prováděno pouze testování programátorem a následovali až akceptační testy u zákazníka. Při rostoucím počtu modelů, funkcionalit a počtu vývojářů byl tento systém již dále neudržitelný a mezi testy programátorem a akceptačními testy musely být vloženy systémové testy prováděny testerem podle testovacích procedur. Dnes, kdy počet modelů routerů přesahuje 30 základních modelů a nové funkcionality přibývají čím dál tím rychleji, je tento systém dále neudržitelný. Již nyní by kompletní testování všech funkcionalit na všech typech routerů trvalo přibližně měsíc práce v jednoho člověka.

Obrázek 1 Příklad routeru

Na základě těchto skutečností byl vznešen požadavek na testovací systém, pomocí kterého bude možné automaticky testovat aktuálně vyvíjený firmware na všech vyráběných modelech routerů a jejich volitelných portech. Zvolena byla kombinace integračního testování a testování založené 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 nepřínosné. Systémové testování by mělo probíhat alespoň jednou denně, aby bylo možno případné chyby odhalit již během vývoje firmwaru a tím se usnadnil a zrychlil samotný vývoj. Dalším velikým přínosem je zkvalitnění samotného testování, jelikož testeři mohou vymýšlet nové testovací procedury a situace namísto opakovaného manuálního testování stejných procedur.

1.2 Aktuálnost

Testování každého výrobku před uvedením na trh, či testováním nového firmwaru před jeho vydáním, je velmi důležitá součást vývoje a neměla by se opomíjet. Hlavním důvodem průběžného testování kompletní funkcionality zařízení je dobré jméno u zákazníka, který nemá zájem o výrobek plný chyb. Dalším důvodem průběžného testování je méně práce pří pozdějším opravování způsobených chyb. Při zvyšování počtu výrobků a funkcionalit je nutné zvyšovat počet testerů nebo změnit přístup k způsobu testování. Jelikož první řešení se zdá být na první pohled neefektivní, tak se v této práci vydám druhou cestou. Při zvyšování efektivity testů použiju automatizované testování. Dále pokud to bude možné, a alespoň trochu efektivní, využiji dnes velmi moderní metodu 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

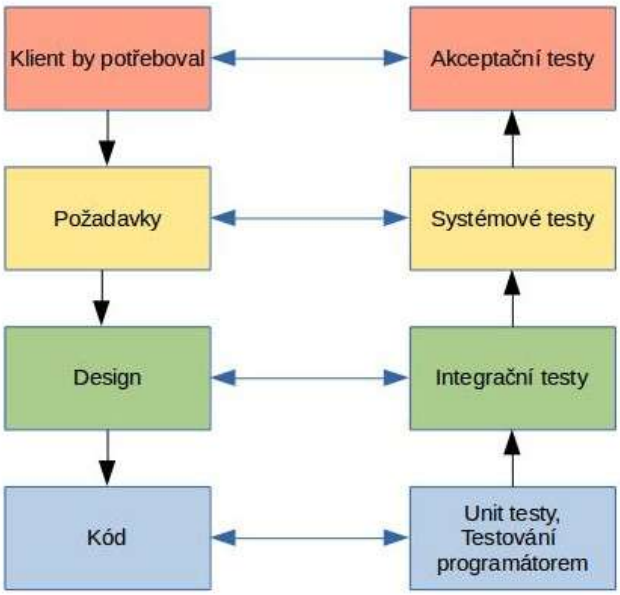
Celou diplomovou práci lze rozdělit do třech hlavních částí. V první teoretické části budou rozebírány všechny různé metody a přístupy k testování samotnému, aby bylo možné dále stavět na teoretickém základu. Dále v teoretické části budou prozkoumány všechny možné dostupné produkty určené k samotnému testování, či produkty sloužící k jednotlivým úkonům 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. Stěžejní částí je kapitola popisující testovací procedury jednotlivých funkcionalit. Poslední částí je praktická implementace testovací laboratoře a testování celého systému. Výstupy 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 systém s ohledem na dnešní metody a trendy v testování. Nejdříve popíšu úrovně testování, kterými by měl každý výrobek před uvedením na trh projít. Dále popíši 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 opomíjeny. Mnou rozebíraný model má celkem 5 stupňů testování. Jednotlivé stupně dále popíši a rozeberu jejich přínos a možnosti použití v mém testovacím systému.



Obrázek 2 Schéma testovacího modelu

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 funkcionalita 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 větve repozitáře projektu. Všechny chyby odchycené v této fázi testování ušetří spoustu času stráveném v dalších fázích testování.

Testování programátorem 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)

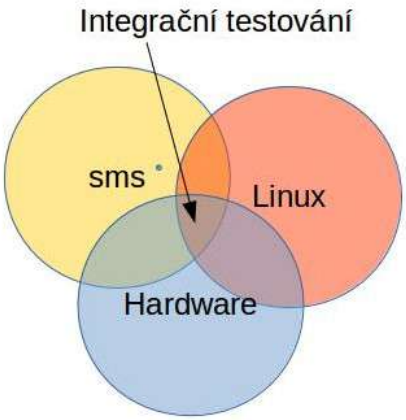
Úroveň testování jednotek obsahuje testování jednotlivých částí nebo modulů softwaru. Za jednotku, neboli část, lze považovat objekt s jednou jedinou funkcionalitou, a to například třídu, objekt, program či softwarový modul. Tato úroveň testování testuje správnost zdrojového kódu a ne funkci programu jako celku. Velmi známým příkladem jsou JUnit testy v javě, 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 neúměrnou námahu a mnoho úprav kódu pro přizpůsobování samotného programu pro unit testování.

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

2.1.3 Integrační testování (Integration testing)

Po předchozích dvou úrovních testování, jenž jsou prováděny programátory, přichází fáze, kdy se hotový výrobek dostává do ruky testerům. Testeři většinou provádí dvě úrovně testování, integrační testování a systémové testování. Někdy jsou tyto dvě fáze spojovány do jedné a nazývají se systémově integrační testování. Obě úrovně 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 operačním systému Linux, který běží na hardwaru konkrétního

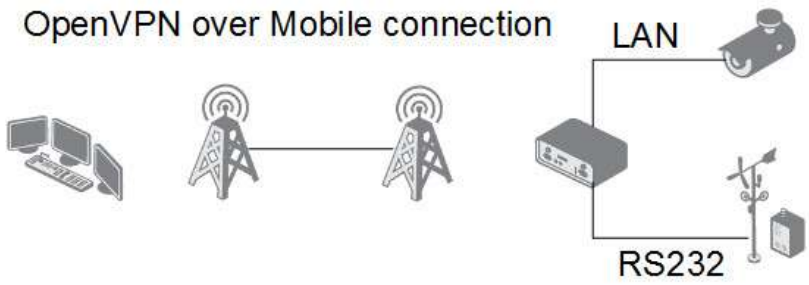
routeru. Příklad znázorňuje, že není testováno pouze odesílání sms zpráv, ale komponenta v závislosti na operačním systému a hardwaru. Jednotlivé komponenty mohou být například subsystémy, databázové implementace, infrastruktura, rozhraní a systémové konfigurace. Integrační testování lze z testování vypustit, jelikož chyby nalezené v této fázi by byly odhaleny ve fázi systémového testování.

Integrační testy navrhují testeři 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 těchto čtyř skutečností tester navrhne testovací případy a postupy. Podle těchto 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 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 chybou některé ze součástí routeru, čímž může být například odlišná implementace odesílání sms v bezdrátovém modulu.

2.1.4 Systémové testování (System testing)

Systémové testování je poslední fází testování během vývoje produktu. Vefázi systémového testování se testuje výrobek jako celek z pohledu zákazníka. Jsou navrhnuty jednotlivé testovací případy, které mohou nastat v praxi, a dle těchto případů jsou výrobky testovány. Příklad takového testovacího scénáře může být testovaný router, do kterého je přes Ethernet připojena IP kamera a přes sériové rozhraní teplotní senzor. Data z těchto zařízení jsou přes OpenVPN tunel, sestavený přes mobilní spojení, posílána na vzdálený server.



Obrázek 4 Příklad systémového testování

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 skříňky. Oba způsoby jsou také dále popsány v kapitole věnující se dalším typům testování.

V navrhovaném 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 hranici 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 nevejde. 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)

Poslední úrovní testování je akceptační testování. Akceptační testování již není prováděno testery ve firmě, kde je výrobek vyvíjen. Testování je prováděno u cílového zákazníka na konkrétní aplikaci výrobku. Případné chyby či nesrovnalosti od požadované funkcionality jsou reportovány zpět a bývá očekávána rychlá reakce na opravu těchto chyb. Snahou testovacího systému samozřejmě bude odhalit všechny případné chyby před touto úrovní, tedy před dodáním výrobku zákazníkovi. Jelikož se tato fáze provádí až u koncového zákazníka, diplomová práce se touto fází dále nebude zabývat.

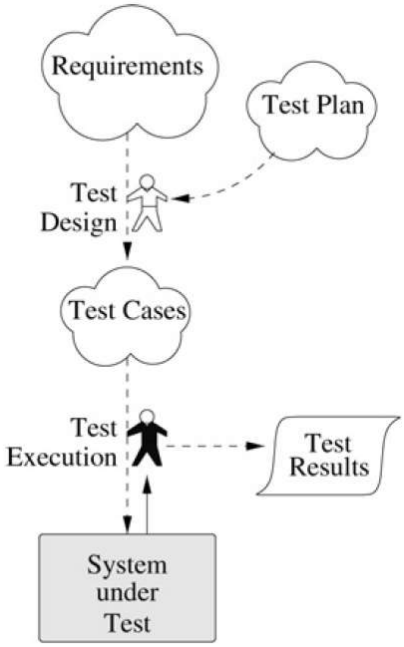
2.2 Testovací procesy

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 popisované 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 možností 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

Prvním přístupem provádění testů je manuální testování. Manuální testování je možné rozdělit do dvou nezávislých kroků. Prvním krokem každého manuálního testování je vytvoření testovacího plánu. Testovací plán obsahuje informace o tom, co by mělo být na výrobku testováno, jak by měl být výrobek testován a nakonec, jak často by měl být výrobek testován. Dle tohoto plánu navrhuje testovací technik testovací scénáře a jejich jednotlivé testovací procedury. Navrhování testovacích procedur se provádí pokaždé při změně nebo přidání funkcionality výrobku. Návrh provádí testovací technik s požadovanými znalostmi o testovaném výrobku.

Druhou fází manuálního testování je samotné provádění testů. Testy se provádějí manuálně přímo na testovaném objektu podle předepsaných testovacích procedur. Provádění testů je opakováno při každé změně ve 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 opakující, 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.



Obrázek 5 Schéma manuálního testování [1]

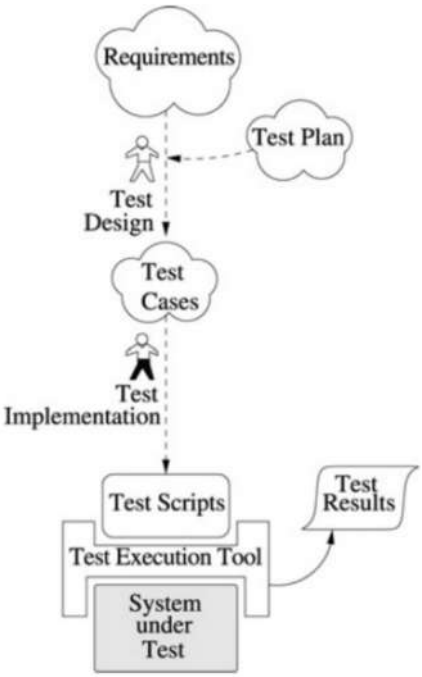
Ve společnosti, kde bude testovací systém nasazován, se nyní všechny testy provádějí manuálně dle předepsaných testovacích procedur. Jak už bylo v úvodu zmíněno, při rychle rostoucím počtu výrobků a jejich funkcionalit je nadále tento systém testování neudržitelný. Provádění integračních a systémových testů se budu snažit 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 zjednodušené manuální 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 vynechání velkého množství testovacích případů, jelikož se jedná spíše o náhodné testování, proto tento postup není určitě doporučován.

2.2.2 Automatizované testování

Druhým, a sofistikovanějším způsobem provádění testů, je automatizované testování, někdy také nazývané testování založené na skriptech. Automatizované testování lze rozdělit do třech 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 či skriptovacím jazyku nebo v jazyku přímo určenému 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 programátora 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



Obrázek 6 Schéma automatického testování [1]

testovacího 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.

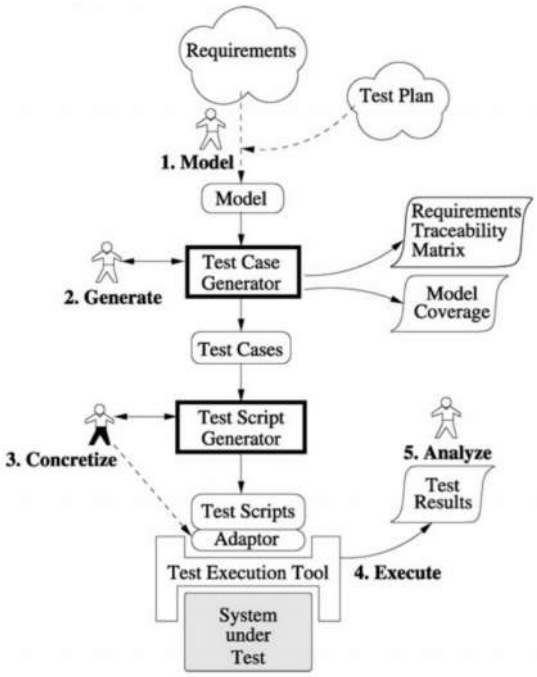
Třetí fází automatického testování je automatické spouštění testů. Testy jsou spouštěny automaticky pomocí nástroje pro spouštění testů. Nástroj provádí spouštění testů automaticky bez interakce s obsluhou. Nástroj navíc obsahuje možnost paralelizace testů nezávislých na nějakém zdroji. Zde je vidět veliká časová a tedy i finanční úspora oproti manuálnímu testování. Jestliže chceme provést nový test zařízení, spustíme pouze testovací nástroj a není potřeba manuálně testovat všechny funkcionality.

Naopak tento přístup přináší větší režii při změně funkcionality či přidání nového zařízení. Pokud je změněna testovací procedura či přímo funkcionalita výrobku, musí být předělány 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 navrhnutí 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 sekvencí operací nad modelem. Používají zjednodušený pohled na testované zařízení a nejsou přímo spustitelné.

Třetí částí testování založeném na modelech je transformace abstraktních testů na spustitelné konkrétní testy. Transformace může být prováděna dvěma způsoby. Prvním způsobem je transformační nástroj, který používá šablony a mapuje každý abstraktní testovací případ do spustitelného skriptu. Nebo je možné napsat adaptér kódu, který implementuje každou abstraktní operaci jako operaci nad testovaném zařízení a doplní jí detaily, které nejsou navrženy v abstraktním modelu.

Ve čtvrtém kroku jsou spouštěny konkrétní testy na testovaném systému. Tato fáze je shodná s třetí fází 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 online 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 příčina a místo vzniku chyby. Nejčastější místa vzniku chyby jsou chybný model, chybný adaptér kódu a v neposlední řadě může chyba vzniknout chybnou 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 ztroskotalo 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 abstraktní 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 nevrací odpověď shodující se s očekávaným výstupem.

2.3.2 Progresní a regresní testy

Progresními testy nazýváme testy kontrolující nové funkce testovaného výrobku. K sestavení progresních testů je nutná znalost nových funkcí daného výrobku. Regresními testy se nazývá opětovné testování již testovaných vlastností výrobků. Regresní testy jsou často prováděny při dokončení části vývoje pro ujištění, zda-li nové úpravy neovlivňují jiné části firmwaru. Taktéž jsou prováděny před vydáním nového firmwaru.

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 systémovými testy. Největší význam těchto testů přichází ve výrobě nových produktů pro ověření funkčnosti vyrobeného výrobků.

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 nesouvisející s jeho funkcionalitou. Jedná se například o testování výkonu celého výrobku nebo jeho částí. 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 skříňky

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

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

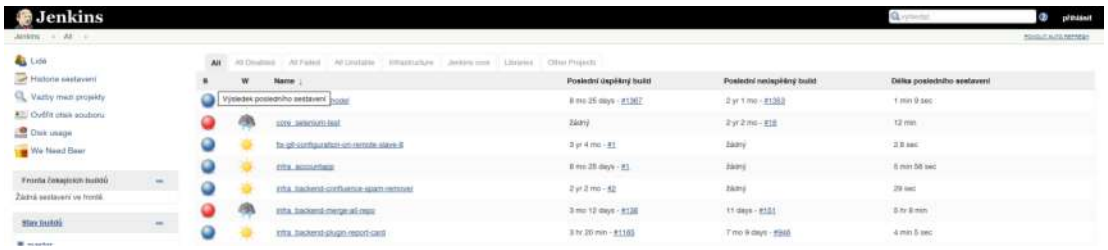
V kapitole dostupné testovací nástroje popíšu zkoumanou část dostupných opensource i komerčních nástrojů pro jednotlivé fáze testování. U všech testovacích nástrojů bude posuzován přínos v případě nasazení v testovacím systému pro výrobky společnosti Conel.

3.1 Jenkins CI

Jenkins CI je nástroj pro kontinuální integraci. Kontinuální integrace spočívá v častém 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 dobrý a přehledný buildovací systém s podporou mnoha programovacích jazyků. Velikou výhodou tohoto systému je opensource řešení projektu, naopak nevýhodou jsou veliké systémové požadavky nástroje. Tento systém sice podporuje testování, ale podporuje pouze testování na úrovní testování jednotek, které v navrhovaném 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 Jenkinsen

Použití nástroje Jenkins CI pro kontinuální integraci v testovacím zařízení by bylo možné pouze ve fázi buildování všech firmwarů. Skripty zajišťující správnou funkci s buildovacím program ltib je potřeba napsat stejné jak při použití systému Jenkins, tak při vlastním spouštění překladu. LTIB je image systém používající se pro překlad firmwaru většiny testovaných výrobků. Taktéž samotné spouštění těchto skriptů bude v testovacím systému již implementováno v části obsluhující spouštění testovacích skriptů. Na základě těchto skutečností jsem upřednostnil použití vlastního spouštění kompilačních skriptů, hlavně z důvodu jednotného systému pro vizualizaci všech informací o průběhu všech fází překladu a testování.

3.2 Testlink

Testlink je nástroj pro správu a organizaci testů. Nástroj je vytvořený 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

Nástroj testlink umí spravovat a organizovat testy a testovací případy. Testlink obsahuje pět základních pilířů, o které se dále opírá a tvoří jeho strukturu. Prvním a základním prvkem je testovací případ neboli test case. Testovací případ lze přirovnat k skriptu či testovací proceduře popisované ve způsobech testování. Dalším prvkem je sada testů označovaná test suite. Sada testů organizuje testy do funkčních skupin. Třetím prvkem nástroje testlink je testovací plán. Testovací plán je sestaven ze sad testů a dále z dalších informací o daném testování. Dalším prvkem je testovací projekt, který je základním prvkem systému. Testovací projekt v známé terminologii odpovídá testovanému subjektu. Posledním prvkem tohoto systému je uživatel. Jednotliví uživatelé mají různá práva úprav v systému, aby byla zavedena bezpečnost systému. Tento systém umí také podle vložených výsledků testů vytvářet reporty o testování v různých formátech.

Obrázek 11 Příklad otevřené aplikace Testlink

Nástroj Testlink může být účinný v případě manuálního testování pro snadnou správu testů a reportování výsledků těchto testů. I v případě navrhovaném testovacím 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 těmto závěrům jsem se rozhodl pro nepoužití nástroje Testlink pro spravování testů a reportování výsledků.

3.3 Selenium

Selenium je nástroj pro testování webových aplikací. Nástroj Selenium je také open-source projekt napsaný v jazyce Java. Selenium je dostupné ve čtyřech variantách a to Selenium IDE, Selenium RC, Selenium WebDriver a Selenium Grid. Selenium IDE je plugin do internetového prohlížeče Firefox. Dále Selenium RC, který si sám pouští internetové prohlížeče a provádí na nich testy. Testy je možné psát v jazycích Java, C, Python, Ruby, Perl, PHP a pro psaní těchto testů je připraveno přehledné API. Seleneium WebDriver usnadňuje psaní samotných testů na úkor nutnosti psaní všech testů

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

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

Selenium je dobrý nástroj pro testování webových aplikací. Nástroj bude určitě využitelný i pro testování webového rozhraní zařízeních testovaných v testovací laboratoři. V první fázi nasazení testovací laboratoře budou jednotlivé konfigurace měněny pomocí ssh a telnet přístupu do routeru. Jelikož tento nástroj více možností nepodporuje, tak nebude v první fázi nasazování testovacího zařízení využit.

3.4 VectorCAST

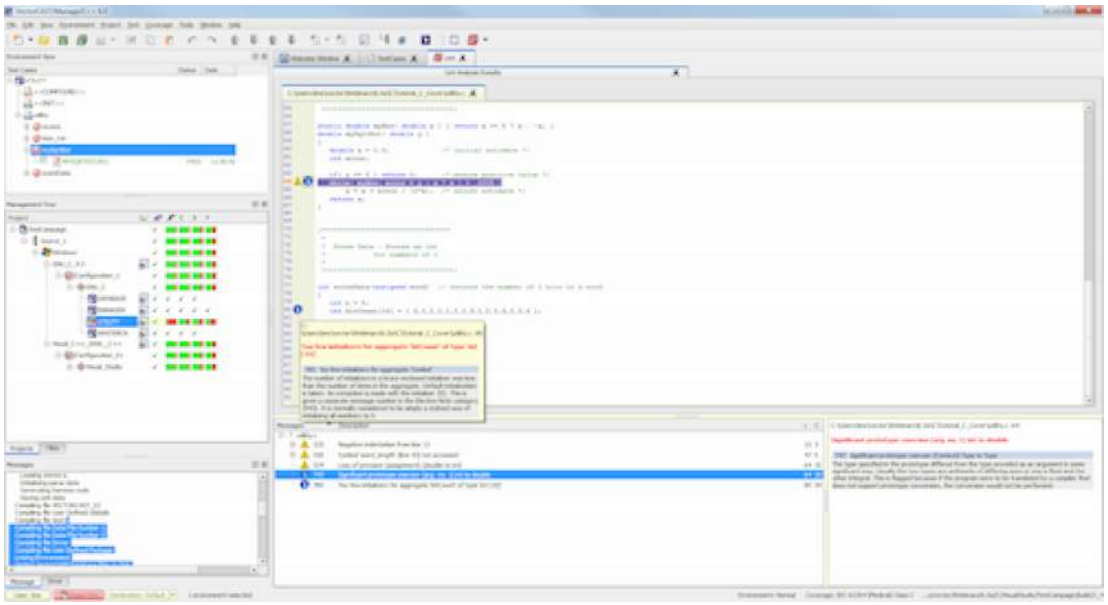
Nejlepším zkoumaným testovacím nástrojem byl komerční produkt VectorCAST od společnosti VECTOR Software. VectorCAST je platforma přímo určená k testování embedded zařízení podporující velkou škálu zařízení a testů.

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

Samotný produkt je rozdělen do několika částí, které jsou dodávány zvlášť podle potřeb zákazníku a typu cílové testované aplikace. První dva programy se zabývají testováním jednotek a integračním testováním. První program je VevtorCAST/Ada, který je určen pro testování softwaru napsaném 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++.

VectorCAST/C++ je řešení pro automatizované testování jednotek a integrační testování. Tento nástroj přímo automatizuje vytváření unit testů pro samotný zdrojový kód. Nástroj je možné spustit nativně nebo na konkrétní cílové platformě. Pro spouštění na cílové platformě je potřeba nástroj VectorCAST/RSP, který bude popsán později. Další vlastností tohoto systému je jednoduché regresní testování softwaru, kdy je možné automaticky spouštět již vytvořené testy. Nástroj může pracovat ve dvou módech a to Source mód a Agile mód. První Source mód tvoří testy z modulů, které jsou již implementovány. Druhý Agile mód nepotřebuje pro tvorbu testů zdrojové kódy daného modulu, ale stačí mu pouze jeho hlavičkové soubory.

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 aplikace 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-



Obrázek 14 Příklad otevřené aplikace VectorCast

ňuje 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.

Testované aplikace je možné testovat přímo na cílové platformě pomocí již dříve zmíněného nástroj VectorCAST/RSP. V tomto nástroji je implementována široká škála kompilátorů. Pomocí VectorCAST/RSP lze stáhnout testovací postup přímo na cílovou platformu, kde jsou všechny testy prováděny a výsledky odeslány zpět testovacímu zařízení. Všechny kroky probíhají automaticky na pozadí bez interaktivity obsluhy.

Řešení pro testování embbeded aplikací VectorCAST, a jeho všechny doplňující balíky, je velmi silným nástrojem. Po detailní zkoumání tohoto nástroje a jeho funkcionalit nemohu ani tento nástroj doporučit pro účely automatické testování výrobků společnosti Conel. Hlavním důvodem nevhodnosti použití tohoto nástroje je potřeba zdrojového kódu k tvorbě jak testování jednotek, tak integračního testování. Firmware v testovaných výrobcích je z devadesáti procent tvořen opensource projekty. 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 testovacímu 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.

```
*** Settings ***
Library          CalculatorLibrary

*** Test Cases ***
Addition
    Given calculator has been cleared
    When user types "1 + 1"
    and user pushes equals
    Then result is "2"

*** Keywords ***
Calculator has been cleared
    Push button    C

User types "${expression}"
    Push buttons    ${expression}

User pushes equals
    Push button    =

Result is "${result}"
    Result should be    ${result}
```

Obrázek 15 Příklad testu ve framewotku http://robotframework.org/

Robot 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 přídavných knihoven a systém upravit k potřebám testovacího zařízení, jednodušší cestou však bude vzít 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 mohl 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.

3.9 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 navrhnu 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 obdélníkovým 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 přidělávat 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 sítí a odděluje testovaná zařízení od firemní sítě. Testovací server může být fyzicky umístěn jinde nežli uvnitř racku, protože ho s testovací sítí 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 diskem. V případě potřeby budou pro archivování starších logů a přeložených firmwarů doplněny další klasické magnetické disky. Pro účely testování bude server osazen dvěmi gigabitovými ethernetovými kartami pro komunikaci s testovanými výrobky a jednou gigabitovou ethernetovou kartou pro připojení serveru do firemní sítě. Server dále má osazeno jedno WiFi rozhraní pro testování WiFi konektivity testovaných výrobků.

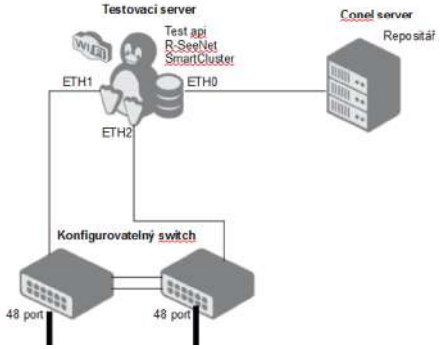
4.1.2 Softwarová výbava testovacího serveru

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 výsledku 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 tvořící 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 kódu 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 switche

Pro připojení všech výrobků k testovacímu serveru budou použity konfigurovatelné 48 portové switche od firmy CISCO. Dva switche byly zvoleny z důvodu velké ceny switchů nad 48 portů. Konfigurovatelné switche budou potřeba z důvodu možnosti změny síťové infrastruktury v průběhu testů bez manuálního zásahu. Každý switch bude připojen k jednomu ethernetovému rozhraní serverua budou propojeny i navzájem. Nejen všechny testované výrobky, ale každé fyzické ethernetové rozhraní každého routeru bude připojeno do switche. Pomocí VLAN bude možné jednotlivé porty stejného zařízení oddělit a vytvořit požadovanou infrastrukturu testovací sítě. Do switche budou dále připojeny další pomocné zařízení použité k testování, například CISCO router pro ověření funkcionality proti jinému referenčnímu zařízení.



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 úrovně výbavy možných rozhraní. Všechny zařízení mají možnost přidělání na DIN lištu a tímto způsobem budou upevněny na testovací stojan.

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 UR5, 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 portů bude popsáno 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 vývoje oddělena, protože je založena na plnohodnotném operačním systému Linux, namísto uCLinuxu použitém u předchozích 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

Řada routerů v2 sebou přináší novou modulární koncepci, a tím i velký počet různých typů routerů. Pro pokrytí všech možných routerů této řady bude muset v testovací laboratoři běžet přibližně třicet routerů. Každý router bude připojen Ethernet kabelem do konfigurovatelného switche. Vybrané routery budou mít zapojen binární vstup a výstup pro testování vstupů a výstupů. Dále tyto routery budou mít zapojeny různá zařízení do USB konektoru, například flash disk nebo USB/RS232 převodník. Modelová řada v2 obsahuje taktéž až dva volitelné porty, popis zapojení testování těchto portů bude popsáno v samostatné kapitole věnující se konkrétním volitelným portům. [4]

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

Modelová řada v3 je poslední vyvinutou řadou routerů. Koncepčně se příliš neliší od předchozí řady. Různé kombinace routerů lze tvořit pomocí dvou volitelných portů. Jednotlivé porty lze osadit jakýmkoliv volitelným portem nebo deskou s bezdrátovým modulem. Aby bylo možné obsáhnout testování nabízených kombinací routerů, bude v testovací laboratoři muset běžet přibližně dvacet routerů této řady. V základní konfiguraci bude muset být navíc proti v2 řadě testován jeden binární vstup a jedno ethernetové rozhraní. [5]

a) Spectre v3 LTE v plastové krabici.

b) Spectre v3 LTE v kovové krabici.

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. Testován 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 stejný 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 přenášena data. Druhým způsobem testů bude připojení loopback desky, 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 pomoci 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

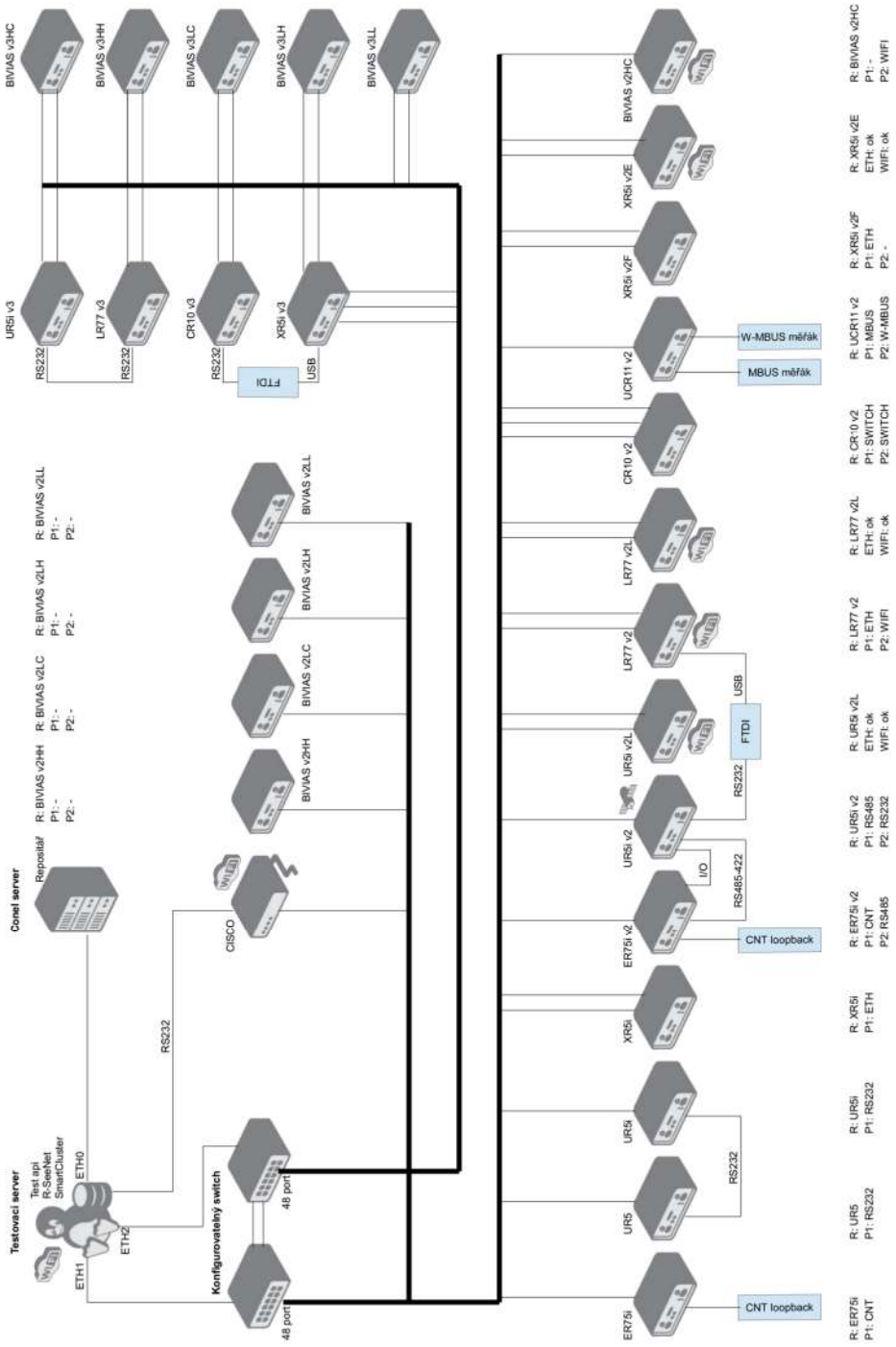
Volitelný port Switch rozšiřuje router o další tři switchované ethernet porty. Po připojení portu Switch do routeru je první ethernet switchován do všech tří portů routeru u modelové řady v2. Modelová řada router v3 po připojení volitelného portu switch obsahuje 3 nezávislé ethernet rozhraní switchované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

4.5 Cisco router

Další důležitou součástí testovací sítě je bezdrátový Cisco router. Router je zapojen do testovací sítě pomocí ethernetu. 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

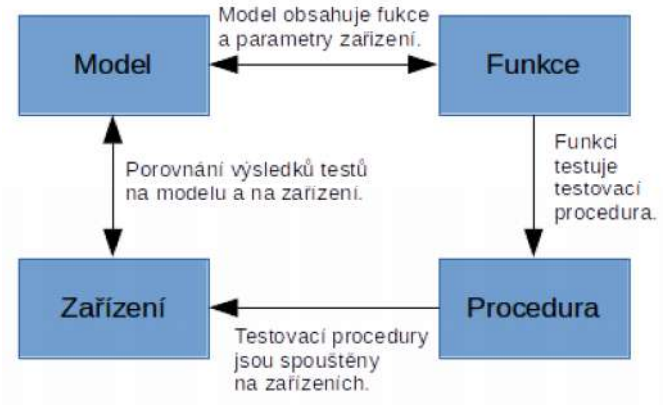
V testovací síti budou dále přibývat různá zařízení pro možnosti lepšího testování jednotlivých rozhraní všech routerů. Mezi tyto zařízení budou například patřit různé měřiče se sériovým rozhraním RS232 nebo M-BUS měřiče. Dále budou v testovací síti umístěny konkurenční výrobky pro ověření funkčnosti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



Obrázek 21 Schéma testovací laboratoře

5 Pohled na modelové testování

Po schromáždění všech požadavků na testování výrobků a na samotnou testovací laboratoř byl navržen způsob implementace testování založeného na modelech. V první fázi nasazování tohoto způsobu testování routerů společnosti Conel bude uvažováno pouze systémové testování všech testovaných zařízení. Modely budou vytvářeny na největší abstraktní míře a to na úrovni jednotlivých funkcí. Zjednodušeně popsáno má každé zařízení definováno svůj model, který nadefinuje správce testovacího zařízení. Každému modelu jsou přiřazeny vlastnosti zařízení a jednotlivé funkce, které by mělo modelované zařízení podporovat. Každá funkce obsahuje sadu spustitelných testovacích procedur, pomocí kterých je skutečné zařízení testováno. Na závěr celého testování se porovnává výsledek spouštěných procedur na skutečných zařízeních s výsledkem testu funkce na modelu zařízení. Na každém zařízení jsou spouštěny testy pouze funkcí, které podporuje model testovaného zařízení. Porovnání provádějí testovací skripty jednotlivých funkcí na základě vlastností modelu zjištěných z databáze a na základě vlastností testovaného zařízení zjištěných z připojeného testovaného zařízení.

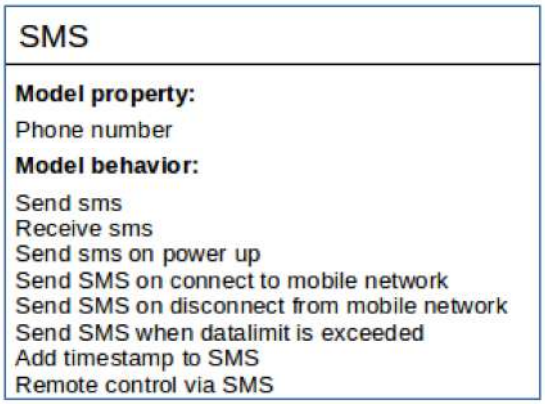


Obrázek 22 Pohled na modelové testování

5.1 Funkce zařízení

Základním pojmem tohoto přístupu bude funkce, kterou dané zařízení může podporovat. Každá funkcionalita jakéhokoliv testovaného zařízení je popsána právě touto základní vlastností. Příkladem takové funkce může být posílání sms zpráv. Každý router vlastnící bezdrátový modul umožňující tuto funkcionalitu bude tuto funkci poskytovat a naopak router bez bezdrátového modulu zřejmě tuto funkcionalitu nebude podporovat.

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. Například nemá smysl testovat nahrání nového firmwaru do výrobku, pokud se nepovede úspěšně přeložit nový firmware.



Obrázek 23 Příklad funkce posílání sms zpráv

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 firmwaru. Skript spouští testy na daném routeru a porovnává výsledky testů spuštěném na routeru s výsledky testů spouštěných 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 vystačím 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.

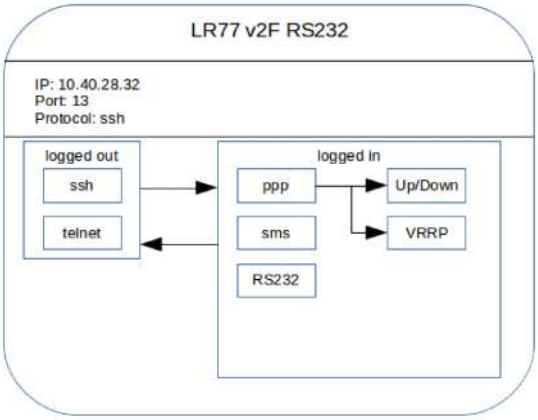
Při testování výrobků testovací program testlab dle parametrů modelu naváže s testovanými výrobky spojení. Po úspěšném navázaní spojení program spouští na daném zařízení testy definované v samotném modelu výrobku. Dále po provedení každého testu program porovnává výsledky testů spuštěných na testovaném 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

Pro snadnější pochopení implementace přístupu testování založené na modelech bude uveden příklad postupu vytváření modelu nového testovaného zařízení a jeho následné testování. Po ukončení základního vývoje je zařízení připojeno do testovací laboratoře. V dalším kroku je potřeba vytvořit model zařízení v testovací aplikaci. Model daného zařízení je vytvořen pomocí webové aplikace, jenž bude detailně popsána v kapitole věnující se webovému rozhraní testovací aplikace. Při vytváření modelu zařízení zadáme jeho parametry. Dále vytvořenému modelu přiřadíme funkce, které testované zařízení podporuje. Tímto krokem je ukončeno přidávání nového výrobku do automatického testování.

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.

Konkrétním případem může být router LR77 v2F RS232, což je bezdrátový LTE router s volitelným sériovým rozhraním. Tomuto routeru přiřadíme IP adresu na primárním LAN rozhraní 10.40.28.32, připojíme ho do třináctého portu switche. Výrobek podporuje komunikaci protokolem SSH i telnet a jako výchozí vybereme SSH protokol. Dále jsou routeru přiřazeny následující funkce, SSH připojení a telnet připojení jsou funkce potřebné k připojení k routeru, pokud tedy testování těchto funkcí skončí úspěšně pokračuje se na testování dalších funkcionalit routeru. Nyní je testováno připojení do mobilní sítě, jestliže je router úspěšně připojen je možné testovat Up/Down skripty routeru a zálohované připojení VRRP. Dále nezávisle na ppp spojení je testována funkcionalita sms zpráv a sériového rozhraní RS232.



Obrázek 24 Příklad modelu routeru LR77 v2F RS232

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 navolených 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

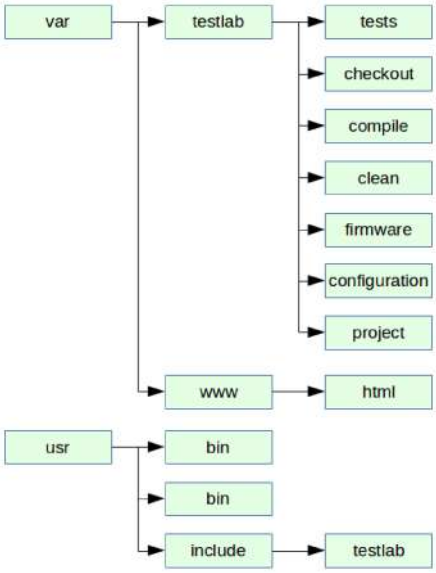
Testovací systém běžící na serveru testovací laboratoře se skládá z několika samostatných částí. Základem celého systému je databáze uchovávající všechny informace o struktuře testovací laboratoře, informace o všech modelech testovaných zařízení a data s výsledky jednotlivých testů. Další součástí je program testlab, jenž se stará o celý průběh testování. Několika málo přepínači lze nastavit průběh testování. Další součástí je sada programů nazývající 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 platforem, 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

Jednotlivé části testovacího systému jsou rozloženy v adresářové struktuře serveru následovně. Všechny součásti testovacího programu testlab jsou umístěny v adresáři /opt/testlab/main a všechny programy testovacího API jsou umístěny v adresáři testovacího systému /opt/testlab/api. Obě zmíněné cesty jsou přidány do systémové proměnné PATH, aby bylo možné programy spouštět odkudkoliv. Sdílené knihovny, které využívá testovací program a programy testovacího API jsou umístěny v adresáři testovacího systému /opt/testlab/lib. Protože sdílené knihovny používá pouze program testlab, hlavičkové soubory nejsou na server umisťovány. Popsané tři adresáře tvořící první část testovacího systému se za běhu nemění a zůstávájí stejné. Jedinou výjimkou je aktualizace testovacího systému, při které mohou být opraveny chyby, či může být přidán nový program do testovacího API. Aktualizaci testovacího programu provádí pouze administrátor 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 platforem. Adresář compile obsahuje skripty zajišťující kompilaci jednotlivých výrobků všech platforem každého projektu. Pro každý projekt je v tomto adresáři sada skriptů, platforma a produkt se zadává jako parametr. V adresáři checkout nalezneme 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. Project je pracovním adresářem, kam jsou kopírovány zdrojové kódy jednotlivých platforem, a kde jsou následně překládány. Dále je tento adresář rozdělen do jednotlivých podadresářů dle jednotlivých verzí překládaného firmwaru. V každém adresáři releasu jsou adresáře pro každou testovanou platformu. Adresáře jednotlivých verzí se před ukončením kompilace odstraňují, jelikož dále nejsou potřeba a zabírají velký prostor na disku. Přeložený 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 firmwaru. 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 databázi. 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

První základní tabulkou celého systému je tabulka fwreleases. V tabulce fwreleases jsou uloženy informace o testovaném releasu. Release je vytvořen a uložen do databáze při každém spuštění programu testlab, aby bylo možné rozlišit jednotlivá testování. Všechny tabulky uchovávající informace o konkrétním testu odkazují právě na jeden konkrétní záznam této tabulky. Tabulka fwreleases prozatím obsahuje pouze 3 údaje.

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

Obrázek 26 Tabulka fwreleases

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 kódu platformy a pro přeložení zdrojových kódů platformy.

Obrázek 27 Tabulka platforms

6.2.3 Tabulka products

Tabulka products obsahuje informace o jednotlivých produktech. V této tabulce nejsou produkty chápány jako jednotlivé produkty v testovací laboratoři, ale pouze jednotlivé druhy firmwaru. Rozdělení bylo provedeno z důvodu možnosti nahrání stejného firmwaru do různého hardwaru a tím vzniknutím jiného výrobku. Tabulka products obsahuje pouze 3 následující položky. Položka idproducts, které je primárním klíčem tabulky, položka idplatforms odkazující na danou platformu v tabulce platforms. Poslední položka name definuje název produktu.

Obrázek 28 Tabulka products

6.2.4 Tabulka checkout

Tabulka checkout slouží pro ukládání výsledků stažení zdrojových kódů z repozitáře. Tabulka obsahuje 4 položky. Položka idcheckout je primárním klíčem tabulky. Položka idplatforms odkazuje na tabulku platforms a udává stahovanou platformu. Položka idfwreleases odkazuje na tabulku fwreleases a udává release, který je testován. Poslední položka state ukládá stav výsledku skriptu pro stažení aktuálních zdrojových kódů platformy.

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 firmwaru je zdrojový kód překládán. Poslední položka state představuje stav ukončení překladu firmwaru celé platformy.

Obrázek 30 Tabulka builds platform

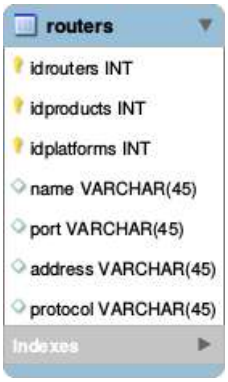
6.2.6 Tabulka build product

Tabulka builds_product slouží k ukládání výsledků překladu jednotlivých výrobků. Tabulka obsahuje celkem 4 položky. Položka idbuilds_product je primárním klíčem tabulky. Položka idproducts odkazuje na tabulku products a udává produkt, jenž je překládán. Položka idfwreleases odkazuje na tabulku fwreleases a udává, k jakému vydání firmwaru je zdrojový kód překládán. Poslední položka state představuje stav ukončení překladu firmwaru daného produktu.

Obrázek 31 Tabulka builds product

6.2.7 Tabulka routers

Tabulka routers je určena pro uložení modelů všech routerů přítomných v testovací laboratoři. Položky v této tabulce tedy nepředstavují již dříve zmíněný produkt, ale skutečný testovaný výrobek. Každá položka této tabulky představuje jeden model testovaného zařízení, tudíž testovány jsou pouze zařízení, které jsou uloženy v této tabulce. Kompletní model zařízení netvoří pouze tabulka routers. Model zařízení tvoří i další tabulky, které jsou s tabulkou routers provázány různými typy vazeb. Tabulka routers obsahuje následující položky. Položka idrouters je primárním klíčem tabulky. Položka idproducts odkazuje do tabulky products a definuje firmware, jenž má být nahrán do tohoto výrobku. Položka idplatform odkazuje do tabulky platforms a definuje platformu, na které je daný výrobek postaven. Další položka name slouží k pojmenování výrobku v testovací laboratoři, tato položka slouží pouze k snadnému rozeznání výrobků v testovací laboratoři. Položka port definuje v jakém portu switche je zapojen primární ethernet testovaného výrobku. Výchozí IP adresu primárního portu testovaného zařízení definuje položka address. Prozatím poslední položkou tabulky routers je protocol. Položka protocol určuje primární protokol, pomocí kterého testovaný výrobek komunikuje s testovací aplikací.



Obrázek 32 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.

Obrázek 33 Tabulka functions

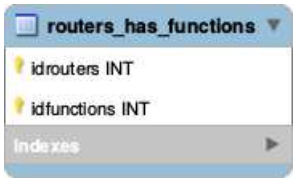
6.2.9 Tabulka dependences

Třetí tabulkou tvořící model testovaného zařízení je tabulka dependences. Pomocí tabulky dependences je možné definovat závislosti jednotlivých funkcí na jiných, tudíž je možné definovat spuštění jednoho testu v závislosti na výsledku jednoho nebo více předešlých testů. Popsaná funkcionalita je realizována pomocí následujících třech položek. Položka target odkazuje na určitou funkci a určuje funkci, která je závislá na výsledku jiných funkcí. Položka dependences dále odkazuje na zdrojovou funkci a určuje funkci, na které závisí provedení testů funkce target. Poslední položka určuje typ závislosti. Podporovány jsou dva typy závislosti, buď musí být správně otestovány všechny předešlé funkce nebo stačí úspěšný výsledek testů pouze jedné ze zdrojových funkcí.

Obrázek 34 Tabulka dependences

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.

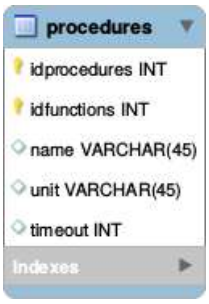


Obrázek 35 Tabulka routers has functions

6.2.11 Tabulka procedures

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

k testování funkcí. Procedury jsou definované následujícími položkami. Položka idprocedures je primárním klíčem tabulky. Položka idfunctions odkazuje na záznam v tabulce functions a definuje funkci, kterou má procedura testovat. Položka name definuje název procedury, pod kterým se spustitelný skript v testovacím systému reprezentuje. Položka unit slouží k lepší reprezentaci výsledné hodnoty daného skriptu. Pokud procedura vrací hodnotu, která je definována jakoukoliv jednotkou, může být v položce unit tato jednotka definována. Položka timeout určuje čas maximálního běhu testovací procedury. Po uplynutí tohoto času je testovací procedura ukončena.



Obrázek 36 Tabulka procedures

6.2.12 Tabulka tests router

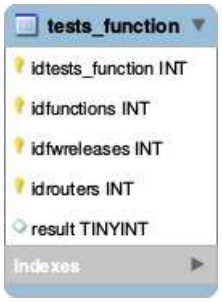
Výsledky testování jsou ukládány do tří různých tabulek pro jednodušší reportování výsledků testování. První tabulkou je tests_router do níž jsou ukládány informace o testování celého zařízení. Za úspěšný se tento test považuje, pokud všechny procedury spuštěné na testovaném zařízení byly ukončeny úspěšně. Tabulka obsahuje čtyři následující položky. Položka idtests_router je primárním klíčem tabulky. Položka idrouters odkazuje na tabulku routers a určuje jakému zařízení je výsledek testu určen. Položka idfwreleases odkazuje na tabulku fwreleases 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

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 proběhli 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 idfwreleases odkazuje na tabulku fwreleases a definuje k jakému testovanému firmwaru je výsledek testování

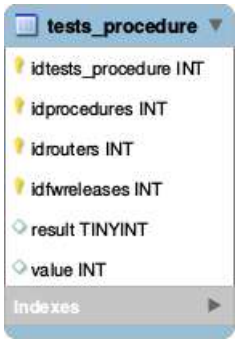
funkce přiřazen. Položka idrouters odkazuje na tabulku routers a definuje jakému zařízení je výsledek testu přiřazen. Poslední položka result určuje samotný výsledek testu.



Obrázek 38 Tabulka tests function

6.2.14 Tabulka tests procedure

Poslední tabulkou sloužící k ukládání výsledků testů je tabulka tests_procedures. V této tabulce jsou uloženy výsledky ze všech spuštěných testovacích procedur. K identifikaci výsledku testu z každé testovací procedury slouží šest následujících položek. Položka idtests_procedure je primárním klíčem tabulky. Položka idprocedures odkazuje na tabulku procedures a definuje testovací proceduru. Položka idrouters odkazuje na tabulku routers a definuje jakému testovanému zařízení má být výsledek přiřazen. Položka idfwreleses odkazuje na tabulku fwreleases a definuje jakému testovanému releasu má být výsledek testu přiřazen. Položka result určuje výsledek testu. Poslední položka value slouží k uložení jakékoliv pomocné informační hodnoty, kterou může vygenerovat testovací skript.



Obrázek 39 Tabulka tests procedure

6.2.15 Tabulka logs

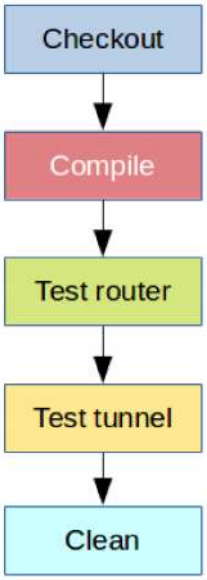
Tabulka logs shromažďuje všechny chybové výpisy, jenž jsou vygenerovány testovacími skripty. Tabulka obsahuje pět následujících položek. Položka idlogs je primárním klíčem tabulky. Položka idprocedures odkazuje na tabulku procedures a definuje k jaké testovací proceduře je chybová hláška přiřazena. Položka idrouters odkazuje na tabulku routers a definuje testované zařízení, kde chyba vznikla. Položka idfwreleases odkazuje na tabulka fwreleases a definuje jakého firmwaru se chybová hláška týká. Poslední položka event určuje samotný text chybové hlášky.

Obrázek 40 Tabulka logs

6.3 Popis programu

O průběh celého testu se stará program testlab. Testlab je program psaný v jazyce C. Program po spuštění otevře systémový log pro možnost logování chyb do systémového logu. Filtrovaný systémový log by měl být později zobrazován na webovém rozhraní testovacího systému. První hláškou do systémového logu je informace o spuštění programu testlab daným uživatelem a v určený čas. Po otevření systémového logu program rozebírá parametry na příkazové řádce. Parametry jsou rozebírány pomocí funkce getopts. Pomocí parametrů lze ovlivnit chování programu testlab.

Jediným povinným parametrem je název testovaného projektu v daném repozitáři. Před tento povinný parametr je možné umístit dva volitelné přepínače upravující chování programu. Přepínač -r určuje, zda přeložený firmware bude označen jako release nebo beta verze. V případě že parametr není zadán, firmware je přeložen jako beta verze. Druhým přepínačem -b s povinným parametrem určíme větev repozitáře, ze které mají být zdrojové kódy kompilovány.



Obrázek 41 Základní schéma testovacího programu

Po rozebrání parametrů se provádějí přípravné kroky pro samotné testování. V prvním kroku je připravena nová verze a následně vložena do databáze. V adresáři project 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.

Program checkout je spouštěn se třemi parametry. Prvním parametrem je identifikační číslo testovaného releasu, druhým parametrem je identifikační číslo stahovaného projektu a posledním parametrem je název stahovaného projektu. Nejdříve je kontrolována existence skriptu provádějící stažení platformy. Stahování zdrojových kódů pomocí skriptu bylo zavedeno z důvodu komplexnosti této operace. Jedním ze složitějších případů checkout skriptu může být stahování zdrojových kódů platformy ze dvou a více různých repozitářů. Po zkontrolování existence skriptu je checkout skript spuštěn. Chybový i standardní výstup skriptu je přesměrován do souboru s logem. Tento soubor se nachází v adresáři logs, dále v podadresáři s názvem identifikačního čísla releasu, kde jsou samotné soubory pojmenované podle názvu stahovaného projektu. U skriptu je dále kontrolované, jestli čas spuštění nepřekračuje nastavený timeout. V případě překročení času spuštění je skript ukončen s chybovým návratovým kódem. Na závěr je výsledek stažení projektu uložen do databáze. V této fázi testování je také zjišťována verze firmwaru a hash posledního commitu zdrojového kódu.

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

6.3.2 Compile

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

Program compile je spuštěn se třemi parametry. Prvním parametrem je identifikační číslo testovaného releasu, druhým parametrem je identifikační číslo překládané platformy a posledním parametrem je název překládané platformy. Program compile nejdříve provede změnu pracovního adresáře na adresář se zdrojovými kódy překládané platformy. Program dále z databáze získá názvy všech produktů patřících pod zpracovávanou platformu. Nyní je provedena kontrola existence skriptu provádějícího samotnou kompilaci zdrojových kódů platformy. Jednotlivé produkty se spouštěnému skriptu předávají jako parametr. V případě existence skriptu pro kompilaci zpracovávané platformy je postupně tento skript spouštěn s parametry všech výrobků. Standardní a chybový výstup spuštěného skriptu je přesměrováván do souboru obdobně jako u programu checkout. Díky uložení všech výstupů jsou informace o překladu firmwarů zpětně dostupné. Doba běhu skriptu je kontrolována, a jestliže přesáhne zadaný timeout, skript je předčasně ukončen s chybovým návratovým kódem. Toto opatření je prováděno z důvodu předejití zaseknutí celého testovacího systému. Každý přeložený firmware je zkopírován z adresáře, kde byl přeložen, do adresáře určeného pro přeložené firmwary aktuálního releasu. Po zkopírování firmwaru je výsledek překladu každého výrobku uložen do databáze. Výsledek přeložení platformy, čili všech výrobků, je na závěr této sekce také uložen do databáze z důvodu snadnější prezentace dat.

6.3.3 Remote server

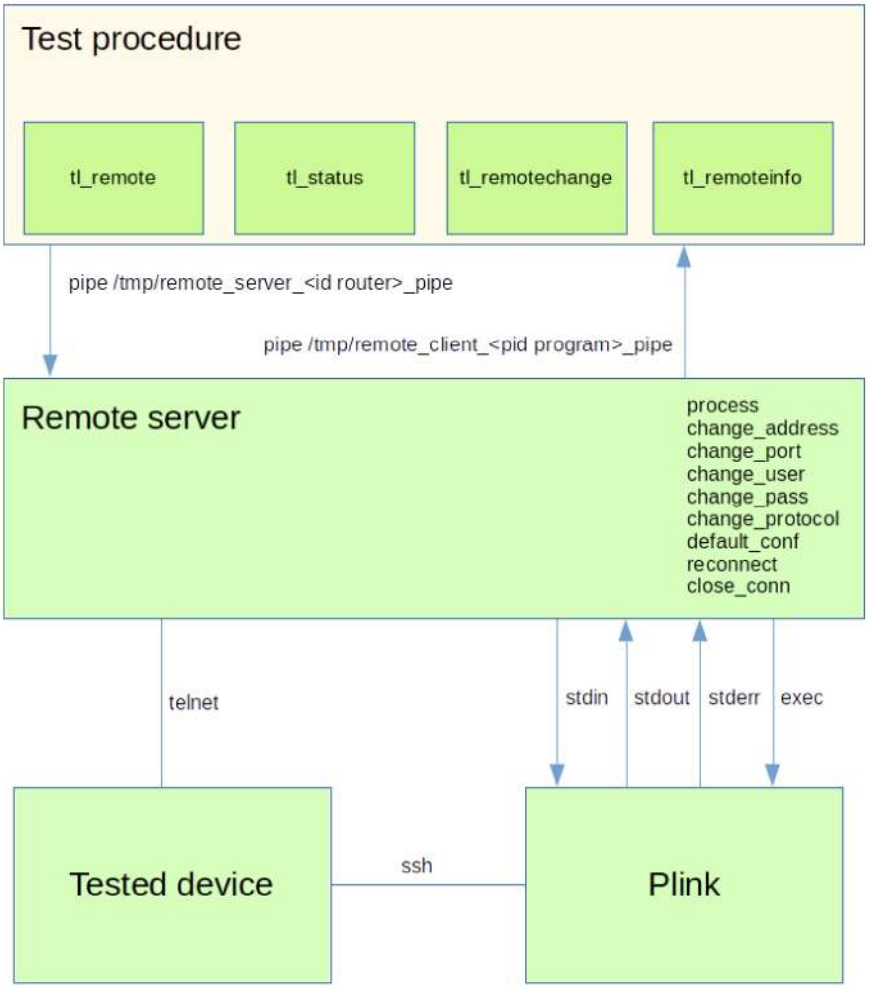
Po ukončení překladu všech výrobků končí fáze kontinuální integrace a začíná systémové testování na konkrétním hardwaru. Pro účely testování se s každým zařízením udržuje permanentní spojení pomocí jednoho z protokolů telnet nebo SSH. Způsob komunikace pomocí permanentního spojení byl zvolen, jelikož 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 je spouštěn se třemi povinnými parametry. Prvním parametrem je identifikátor zařízení. Identifikátor zařízení je číslo, pomocí kterého mohou testovací aplikace komunikovat se zařízením. V rámci testovacího systému je v tomto parametru předáváno identifikační číslo zařízení použité v databázi. Druhým parametrem je IP adresa zařízení s jímž má být udržováno spojení. Posledním povinným parametrem je protokol, kterým je prováděno spojení s zařízením. Nyní jsou podporovány dva protokoly, a to telnet a SSH. Dále je možné použít volitelné parametry k úpravě výchozích přihlašovacích informací. Prvním volitelným parametrem lze změnit přihlašovací jméno do zařízení. Parametr lze změnit pomocí přepínače -u a defaultní hodnota je uživatel root. Dalším volitelný parametrem lze změnit přihlašovací heslo do zařízení. Parametr lze změnit pomocí přepínače -p a jeho defaultní hodnota je přihlašovací heslo root. Posledním volitelný parametrem je port, na kterém se sestavuje spojení s testovaným zařízením. Port lze změnit pomocí přepínače -t a výchozí číslo portu je 23.

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 naslouchá 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 blokovaným 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 žádosti:

- Vykonaní 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 aktualní adrese
- Informace o aktualním portu
- Informace o aktualním uživateli
- Informace o aktualním hesle
- Informace o aktualním 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 jasné z jejich názvů a tak dále bude detailně popsána hlavní a nejvíce používána žádost vykonání příkazu. Samotné vykonání příkazu je provedeno podle zvoleného protokolu.

Telnet

Při provedení příkazu pomocí protokolu telnet je nejdříve kontrolováno, jestli je spojení s testovaným zařízením aktivní. Jestliže spojení aktivní není, pokusí se program spojení s testovaným zařízením znovu navázat. V případě protokolu telnet byla napsána vlastní knihovna pro obsluhování telnet spojení. Navázání spojení je provedeno navázáním TCP spojení na portu, kde testované zařízení obsluhuje příchozí telnet spojení. Po navázání spojení jsou přečteny parametry telnet relace a žádost o zadání přihlašovacího hesla. Nejdříve jsou zkontrolovány parametry relace a následně odeslána odpověď s parametry požadovaného chování telnet relace. Dále je odesláno přihlašovací jméno uživatele routeru. Po odeslání uživatelského jména je očekávána žádost o uživatelské heslo, které je také odesláno pomocí TCP spojení do zařízení. V případě úspěšného přihlášení navazovací funkce vrací číslo file deskriptoru spojení. V případě jakéhokoliv

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 mezera. Pro zamezení chyb při příjmu z routeru je nastaven timeout 60 s. Po přečtení celé odpovědi z routeru je odpověď zpracována a pomocí pojmenované roury odeslána dotazovanému programu. Ukončení telnet spojení s testovaným zařízením je prováděno uzavřením daného TCP spojení.

SSH

Komunikace s testovaným zařízení 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 Pink místo standartního 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 inicializační funkce vrací PID programu Plink. Pokud spojení nebylo úspěšně navázáno, program Plink je ukončen a funkce vrací informaci o neúspěšném připojení.

Po úspěšném přihlášení je možné pomocí remote serveru pomocí SSH protokolu posílat do testovaného zařízení příkazy. Při odesílání příkazů je nejdříve zkontrolováno sestavené spojení. Samotné odesílání příkazů je provedeno skrz nepojmenovanou rouru na standardní vstup aplikace Plink. Následně je čten standardní výstup aplikace Plink do přečtení stejné sekvence znaků jako u telnet spojení. Z důvodu zamezení chyb při přenosu je pro příjem odpovědi nastaven timeout 60 s. Pokud do vypršení timeoutu není příkaz zpracován, spojení je uzavřeno a znovu inicializováno. Po příjmu celé odpovědi je odpověď zpracována a odeslána přes pojmenovanou rouru dotazovanému programu. Ukončení SSH spojení je prováděno posláním signálu kill programu Plink.

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. Třetím 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 releasu. Druhým parametrem programu test je identifikační číslo testovaného zařízení. Po kontrole 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 připravené pro postupné spuštění všech testovacích procedur.

Před samotným spuštění je nejdříve kontrolována existence testovacího skriptu. Dále jsou vytvořeny nepojmenované roury pro předávání informací ze skriptu do testovacího programu. K předávání výsledné hodnoty testu je na rouru 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 nevypršel 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 chybový 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 platforem. 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 ltibu volá pouze ltib -m distclean. Průběh tohoto skriptu je obdobně jako u skriptu checkout a compile logován do souborů v adresáři logs. Po vyčištění všech platforem 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, úkon. Název každého programu se skládá s předpony tl_ a názvu daného programu, který většinou vystihuje jeho účel. Hlavním cílem vytvoření 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 Bashi je možné využít pro komunikaci s remote serverem program tl_remote.

7.1 Knihovna cspipe

Knihovna libtl_cspipe.so slouží ke komunikaci s testovaným zařízení prostřednictvím remote serveru. Knihovna definuje názvy pojmenovaných rour, prostřednictvím kterých je komunikováno s remote serverem, dále velikost příjímacího a odesílacího bufferu a výčet typů všech příkazů, které remote server podporuje. Pro výměnu dat přes roury slouží struktura message_remote, která obsahuje process id dotazujícího 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 pomoví funkce client_starting. Funkci se jako parametr předává process 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 funkcí pipe_request, které se jako prametr předává typ žádosti, buffer s obsahem předávané zprávy a buffer pro naplnění 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

Knihovna database poskytuje funkce pro přístup a manipulaci s databází testovacího systému. Knihovna definuje přihlašovací údaje k samotné databázi a strukturu jednotlivých tabulek. Knihovna také obsahuje funkce pro připojení k databází a následně také odpojení od databáze. K dispozici je řada funkcí, pomocí kterých je možné získat jednotlivá data z databáze. Data jsou z databáze předávány jako dynamicky alokované vícerozměrné pole. Dále je možné využít funkce pro vkládání a úpravu záznamů v databázi. Databázové funkce primárně slouží k využití v hlavním testovacím programu, ale i v některých API programech najde tato knihovna využití.

7.3 Knihovna utils

Poslední vlastní používanou knihovnou je knihovna utils. Knihovna utlis obsahuje funkce pro změny v systému. První funkce close_all_fds slouží k uzavření všech otevření souborových deskriptorů. Druhá a zatím poslední funkce této knihovny zajišťuje 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 parametr 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 zařízením 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

Pomocí programu remotechange je možné změnit parametry připojení a přihlášení do testovaného zařízení. Syntaxe programu je `tl_remotechange [-p <port>] [-i <ip>] [-u <user>] [-s <pass>] [-t <protocol>] <id>`. Parametrem port je změněn TCP port na kterém se pokoušíme připojit. Parametr IP změní cílovou IP adresu připojení. Parametry user a pass určují přihlašovací údaje, pomocí kterých se do testovaného zařízení přihlašujeme. Parametr protocol určuje protokol připojení k testovanému zařízení, nyní jsou k dispozici protokoly telnet a ssh. Posledním povinným parametrem id určíme zařízení, kterému chceme změnit parametry připojení. Program remotechange po startu nejdříve odešle informace o změně parametrů připojení remote serveru, a poté je odeslán remote serveru pokyn k restartu přípojení k testovanému zařízení.

7.7 Program remoteinfo

Program remoteinfo slouží k získání aktuálních informacích 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 přepínače -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

Program routerready čeká na start testovaného zařízení, například po rebootu zařízení. Syntaxe programu je `tl_routerready [-t <timeout>] <id>`. Prvním parametrem timeout lze zadat maximální čas čekání na start routeru, kdy defaultní čas je 120 s. Dále je potřeba zadat parametr id, což je identifikační číslo testovaného zařízení. V první fázi čekání na naběhnutí zařízení je zkoušen ping na testované zařízení. Jestliže do vypršení timeotu byla přijata odpověď, je pokračováno druhou fází testu. V druhé fázi testu je zkoušeno připojení do testovaného zařízení pomocí remote serveru. Jestliže do vypršení timeoutu přijde od remote serveru validní odpověď, je vypsán celkový čas čekání na start routeru a program ukončen. V případě vypršení timeoutu v jakékoliv fázi je vypsána chybová hláška a program je ukončen s návratovým kódem 2.

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 zadáno identifikační čislo zařízení, ze kterého má být status vyčten. Druhým parametrem category určíme požadovanou kategorii statusu, například lan pro informace o síťovém rozhraní. Poslední volitelnou položkou subcategory určíme jakou položku v dané kategorii má program vytisknout. U již zmíněné kategorie lan si můžeme nechat vrátit například položku MAC Address.

7.10 Program updateconf

Pro nahrání nové konfigurace do testovaného zařízení slouží program updateconf. Syntaxe programu je `tl_updateconf -f <config> -t <function> [-d <confdir>] <id>`. Prvním parametrem config je určen název konfigurace, jenž má být do routeru nahrávána. Parametr function určuje testovanou funkci nahrávané konfigurace. Třetím volitelným parametrem confdir je možné změnit adresář s nahrávanými konfiguracemi. Výchozí adresář je /var/testlab/conf. Poslední parametr id určuje router, do kterého má být nová konfigurace nahrána. V první fázi jsou z remote serveru příslušného zařízení zjištěny přístupové a přihlašovací údaje k testovanému zařízení. Dále je pomocí programu curl nahrána nová konfigurace do testovaného zařízení. V případě jakékoliv chyby je aplikace ukončena se stejným návratovým kódem jakým byl ukončen program curl.

7.11 Program updatefw

Nahrávání nového firmwaru 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

má být firmware vybrán. Parametr firmware určuje název nahrávaného firmwaru. Třetí volitelný parametr fwdir umožňuje změnit adresář, odkud jsou firmwary vybírány. Defaultní adresář je /var/testlab/firmware. Posledním parametrem id určíme do jakého testovaného zařízení má být nový firmware nahrán. Při nahrávání nového firmwaru jsou nejdříve z remote server příslušného zařízení zjištěny přístupové a přihlašovací údaje testovaného zařízení. Dále pomocí programu curl je nahrán do zařízení nový firmware, po nahrání nového firmwaru do zařízení se zařízení začne automaticky aktualizovat. Program updatefw je ukončen se stejným návratovým kódem jako program nahrávající nový firmware curl.

7.12 Program slog

Programem slog je možné vyčíst a filtrovat systémový log testovaného zařízení. Syntaxe programu je následující `tl_slog [-n <lines>] [-p <program>] <id>`. Prvním volitelný parametrem upravujeme počet vypisovaných řádků systémového logu. Defaultně je vypisováno sto řádků logu. Druhým také volitelný parametrem je možné filtrovat vypisované řádky logu, například podle spuštěného programu. Posledním parametrem id je určeno, z jakého testovaného zařízení je systémový log vyčítán. Program po spuštění provádí pouze vyčítání systémového logu prostřednictvím remote serveru z testovaného zařízení a případně systémový log dále filtruje dle zadaného parametru.

7.13 Program rlog

Programem rlog je možné vyčítat a filtrovat reboot log testovaného zařízení. Reboot log obsahuje důvody, kvůli jakým bylo testované zařízení v minulosti restartováno. Syntaxe programu rlog je `tl_rlog [-p <program>] <id>`. Pomocí prvního parametru program je možné filtrovat vypisované řádky reboot logu. Druhý parametr id určuje testované zařízení, ze kterého je reboot log vyčítán. Program rlog po spuštění pomocí remote server pouze vyčte z testovaného zařízení reboot log a případně filtruje řádky podle zadaného parametru.

7.14 Program klog

Posledním ze sady programů určených pro vyčítání logů z testovaných zařízeních je je program klog. Programem klog je vyčítán kernel log testovaného zařízení. Syntaxe tohoto programu je `tl_klog [-p <program>] <id>`. Prvním volitelným parametrem lze nastavit filtrování vypisovaných řádků kernel logu. Parametr id určuje testované zařízení, ze kterého bude kernel log vyčítán. Program klog po spuštění pomocí remote server přes príkaz dmesg vyčte z testovaného zařízení kernel log a dále ho případně filtruje podle zadaného parametru, či přímo tiskne na standardní výstup.

7.15 Program changeparam

Program changeparam slouží k provedení změny jednotlivých položek konfigurace testovaného zařízení. Změna se projeví ihned po provedení příkazu a není nutné žádné nahrávání konfigurace do zařízení. Syntaxe programu remotechange 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

parametrem param je určen měněný parametr. U funkce ppp můžeme změnit například parametr apn. Volitelným parametrem profile je možné určit jiný profil konfigurace, kde bude parametr měněn. Dalším parametrem id zvolíme testované zařízení, kterému má být určený parametr změněn. Poslední parametr value udává novou hodnotu měněného parametru. Program changeparam po kontrole všech parametrů převádí názvy těchto parametrů na velká a malá písmena dle potřeby testovaných zařízení. Dále je sestaven příkaz pro změnu parametru v testovaném zařízení pomocí programu sed a následně je pomocí remote server odeslán do testovaného zařízení.

7.16 Program mobileready

Program mobilready slouží k čekání na navázání mobilního spojení testovaného zařízení. Syntaxe programu je `tl_mobileready [-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 prováděno každou sekundu až do vypršení timeoutu nebo do úspěšného navázaní spojení.

8 Uživatelský interface

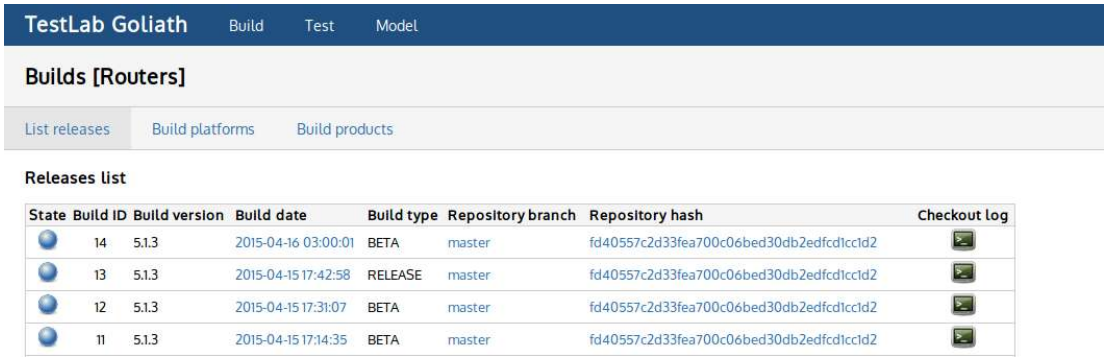
Komunikaci testlabu s jeho uživateli obsluhuje webová aplikace. Webová aplikace je vytvořena pomocí technologií html, css, php a javascript. Pomocí webové aplikaci bude možné prohlížet modely všech zařízení a jednotlivé modely upravovat. Model zařízení obsahuje všechny informace o testovaném zařízení, 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 nalezneme 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.



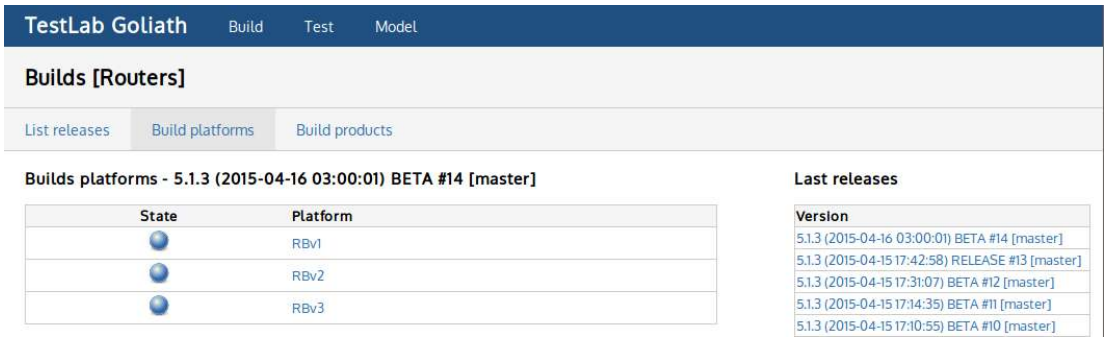
Obrázek 43 Stránka release sekce build

Tabulka má celkem osm sloupců. První sloupec state informuje o výsledku stažení nebo aktualizaci zdrojových kódu 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 identifikující 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

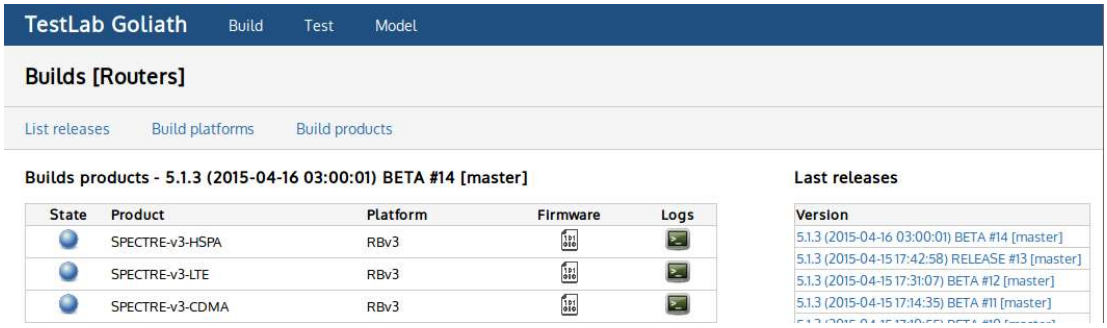
Builds platform je přehledová stránka zobrazující výsledky překladů testovaných platforem. Aktuální zobrazovaný release je zobrazen v nadpisu této stránky. Dále stránka build platforms obsahuje dvě tabulky. První dominantní tabulka zobrazuje výsledky překladů a obsahuje pouze dva sloupce. První sloupec state zobrazuje výsledek překladu platformy a druhý sloupec platform obsahuje názvy překládaných platforem. Název platformy slouží také jako odkaz na přehled překladů všech výrobků této platformy. Druhá menší tabulka obsahuje pouze jeden sloupec se seznamem odkazů na překlady platforem posledních releasů.



Obrázek 44 Stránka platform sekce build

8.1.3 Stránka Build products

Stránka build products již zobrazuje konkrétní informace o překladu vybraných produktů. Stránka má totožné rozložení s předchozí stránkou build platforms. Zde první tabulka informuje o překladech produktů a obsahuje celkem pět sloupců. První sloupec state indikuje výsledek překladu firmwaru. Sloupec product obsahuje název firmwaru překládaného produktu. Sloupec platform zobrazuje platformu daného produktu. V případě úspěšného přeložení firmwaru 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.



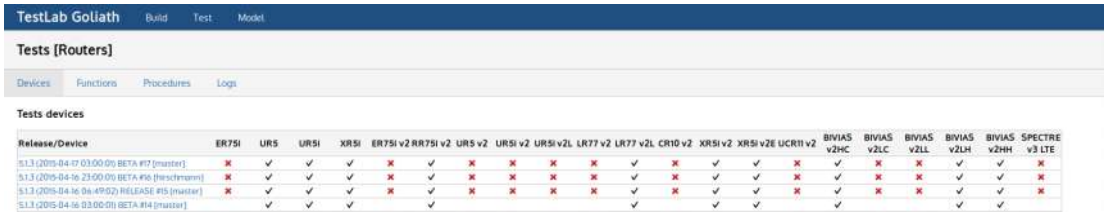
Obrázek 45 Stránka product sekce build

8.2 Sekce test

Sekce test shromažďuje všechny informace z druhé fáze testování, kdy je přeložený firmware nahrán do jednotlivých zařízení, a ty jsou následně testovány. Sekce test obsahuje celkem čtyři stránky. První stránka devices zobrazuje přehled testování jednotlivých zařízení v každé verzi firmwaru. Stránka functions zobrazuje přehled testování funkcí na jednotlivých zařízeních. Stránka procedure zobrazuje výsledky zvolených testovacích procedur na jednotlivých zařízeních. Poslední stránka log vypisuje chybové hlášky vzniklé při průběhu testování.

8.2.1 Stránka Devices

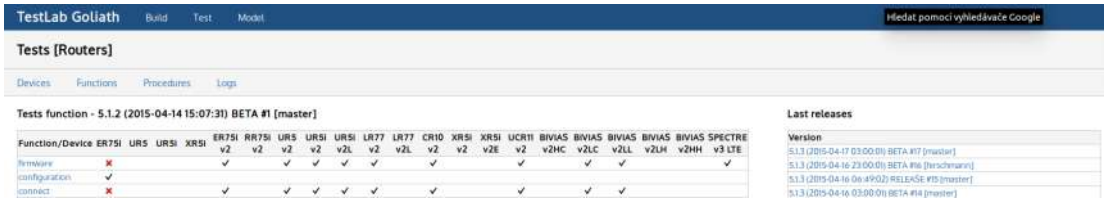
První stránka devices sekce test obsahuje pouze jednu tabulku. V jednotlivých sloupcích tabulky jsou všechna testovaná zařízení a každý řádek obsahuje informace o jedné testované verzi firmwaru. Každá buňka uvnitř tabulky zobrazuje informaci a výsledku průběhu testů na zařízení v určité verzi firmwaru. Výsledek testů může být buď úspěšný, což signalizuje zelená fajfka, nebo může být neúspěšný, což signalizuje červený křížek. Červený křížek také slouží jako odkaz na logy neúspěšných testů odpovídajícího zařízení. V případě že, zařízení nebylo testováno, zůstává buňka prázdná.



Obrázek 46 Stránka product sekce build

8.2.2 Stránka Functions

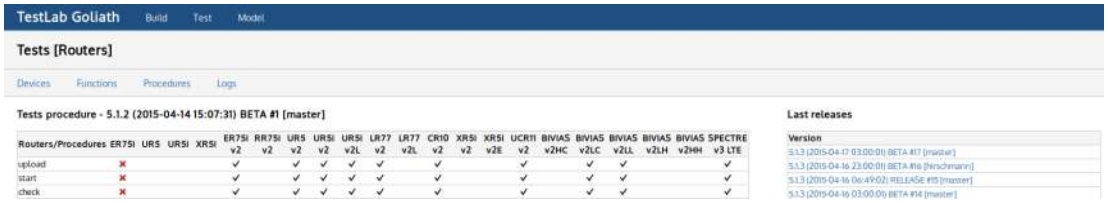
Stránka functions slouží k zobrazení přehledu testů jedné testované verze firmwaru. Verzi firmwaru je možné si vybrat pomocí pravé tabulky obdobně jako na stránce builds platform nebo přechodem z předchozí stránky devices. Informace o aktuálním zobrazeném releasu jsou umístěny v nadpisu stránky. V hlavní tabulce jsou zobrazeny výsledky testů funkcí na jednotlivých zařízeních. Každý sloupec představuje jedno zařízení, každý řádek představuje jednu funkci a všechny buňky uvnitř tabulky zobrazují výsledek testu funkce na daném routeru. Ikona zobrazující neúspěšný test je zároveň odkaz na výpis logu neúspěšných procedur odpovídající funkce.



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

Stránka logs je prozatím poslední stránkou sekce test. Na této stránce jsou zobrazeny všechny chybové hlášky z testovacích procedur. Stránka obsahuje pouze jednu tabulku, kde každý řádek odpovídá jedné chybové hlášce. Tabulka obsahuje celkem pět sloupců. V prvním sloupci release je zobrazen k jaké testované verzi firmwaru chybová hláška odpovídá. Sloupec router určuje testované zařízení, kde chyba vznikla. Místo, z kterého chybová hláška vzešla, určují následující sloupce function a procedure. Poslední sloupec event obsahuje samotnou hlášku předanou chybovým výstupem z testovací procedury.

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 navrhnutou 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.

V neposlední řadě je navrhnut přístup testování založený na modelech. Pomocí tohoto přístupu rozdělím všechny funkcionality všech routerů do funkcí. Následně každou funkci rozdělím na jednotlivé procedury, kdy každá procedura testuje elementární chování této funkce. Například pokud je dána funkce připojení routeru do mobilní sítě, tak v této funkci je testovací procedura testující změny APN tohoto připojení. Pomocí těchto funkcí je možné namodelovat všechny testovaná zařízení. Pomocí těchto modelů jsou pro každé zařízení vybírány a spouštěny konkrétní testy.

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 níže popisované procedury.

Testovací skripty jsou spouštěny se dvěma parametry. Prvním parametre 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

Testování každého projektu je započato stažením zdrojových kódů daného projektu. Způsob stažení zdrojového kódu je zajištěno pomocí skriptů checkout. V tomto skriptu definujeme způsob stažení zdrojových kódu. Například stažení pomocí verzovacích systémů git, či cvs nebo zkopírování projektu z určitého místa. Všechny projekty testovacího systému se stahují pomocí verzovacího systému git přímo ze serveru testovací laboratoře, kde jsou vytvořeny kopie repozitářů všech testovaných projektů. Kopie je pravidelně každou minutu udržována aktuální. Tento postup byl zaveden z důvodu zrychlení samotného testování a rapidního zmenšení datových toků. Primárně je stahována hlavní větev master.

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 ltib, tudíž skripty zajišťující překlad projektů vypadají velice obdobně. Každý

skript je prováděn v adresáři, kde je projekt stažen. V prvním kroku je otevřen adresář ltib. 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 buildovacího systému doplněn v rámci testovací laboratoře, jelikož ltib nepodporoval překládání bez interakce uživatele. Výběr platformy je dokončen spuštěním programu ltib v konfiguračním 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 závisející na výrobku a spuštěn samotný překlad. V případě úspěšného ukončení překladu jsou přeložené firmwary nakopírovány do adresáře předaného jako parametr skriptu. Překládání projektů pomocí skriptů bylo zvoleno z důvodu univerzálnosti 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 prováděn odlišným způsobem, je pro každý projekt vytvořen vlastní skript. Pro projekty, které jsou zatím testovány v testovací laboratoři je nejdříve otevře adresář ltib. Samotný úklid po překladu je proveden spuštěním programu ltib v módu distclean. Na závěr se skript vrátí do výchozího adresáře a je ukončen s návratovým kódem programu ltib.

9.4 Funkce firmware

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

9.4.1 Procedura upload

První testovací procedura funkce firmware testuje nahrání firmwaru do testovaného zařízení. V prvním kroku je zjištěn název firmwaru testovaného zařízení z databáze testovacího systému. Dále je správný firmware nahrán do testovaného zařízení pomocí utility updatefw 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 firmwaru 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 starý firmware. Dále je spuštěn program testující připojení routeru do testovací sítě. V tomto programu je nejdříve testován ping na router a v případě úspěšného pingu je testováno odezva routeru na příkaz echo. Test končí úspěšně, pokud se na routeru podaří ping, a také se k němu podaří připojit jedním z podporovaných protokolů. Tento test vrací také čas startu routeru v sekundách.

9.4.3 Procedura check

Poslední procedura funkce firmware check testuje shodu verzi firmwaru v routeru s nahrávanou verzí firmwaru testovacím systémem. V první fázi je z databáze zjištěn název firmwaru testovaného výrobku. Pomocí názvu firmwaru je nalezen soubor s verzí firmwaru dodávaný s přeloženým firmware a jeho obsah uložen do proměnné. Po zjištění požadovaného firmwaru 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

Další funkcionalitou, kterou podporují všechny testované zařízení je funkcionalita obsluhující konfigurace. Konfigurace zařízení je seznam parametrů s hodnotami určující chování daného zařízení. Funkcionalita konfigurace je také velmi důležitá, jelikož v každém testu je nastavována odlišná konfigurace a sleduje se chování daného zařízení. Pro práci s konfigurací je v zařízeních několik základních nástrojů. Konfigurace je možné zálohovat a nahrávat nové. Dále je možné pracovat s takzvanými profily. K dispozici jsou až čtyři alternativní profily. Nakonec je možné měnit parametry přímo v routeru přepisováním určitých souborů.

9.6 Funkce connect

Nyní přicházejí dvě základní funkcionality, které obstarávají připojení do routeru. K dispozici jsou dvě možnosti připojení do testovaných zařízení. Ne všechna testovaná zařízení podporují oba typy připojení. První typ připojení do routeru je pomocí klasických nešifrovaných protokolů telnet, ftp a http. Jedná se o nezabezpečené připojení, takže není podporováno nejnovějšími zařízení platformy v3.

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 opravdu 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 prováděna porovnáním vráceného řetězce s řetězcem vypisovaným příkazem echo. Posledním krokem tohoto testu je vrácení původního komunikačního 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 souvisejí i další šifrované typy připojení klasických protokolů. Například zabezpečená varianta ftps protokolu pro přenos dat ftp, dále šifrovaná varianta https webové 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 rozdílem, 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 zařízeních 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 přiřazení IP adresy, zkouška komunikace či změna parametru určující velikost odchozího 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 routeru připojení pomocí mobilního spojení. Úspěšné připojení do mobilní sítě je definováno přiřazením IP adresy rozhraní mobilního spojení. Samotný test probíhá pouze spuštěním programu mobileready, který čeká tři minuty na přiřazení IP adresy rozhraní mobilního spojení. Program, tedy i skript, vypisuje pouze čas v sekundách, který čekal na připojení routeru do mobilní sítě. Skript je ukončen úspěšně, pokud se routeru podaří připojit do mobilní sítě do tří minut.

9.8.2 Procedura ping

Procedura ping popisuje vlastnost routeru základní komunikace v mobilní síti. Vlastnost základní komunikace je popsána provedením alespoň jednoho úspěšného pingu na zařízení v mobilní síti. Samotný test probíhá následovně. V prvním kroku je zálohováno původní APN. Poté je změněno APN mobilního spojení na conel.agnep.cz, aby byla na všemi zařízeními přístupná referenční adresa. Kvůli projevení změny APN je proveden restart služby PPP starající se o mobilní spojení. Po restartu mobilního spojení se čeká na sestavení mobilního spojení s novým APN pomocí programu mobileready. V případě úspěšného navázání mobilního spojení je proveden ping s pěti pokusy na adresu 10.0.0.1. Na závěr je parametru APN navrácená původní hodnota. Skript vypisuje číselnou hodnotu v procentech počtu úspěšně provedených pingů. Test je ukončen jako úspěšný, jestliže byl úspěšně přijat zpět alespoň jeden pokus o ping na adresu 10.0.0.1.

9.8.3 Procedura apn

Procedura apn popisuje možnost ovlivnění chování mobilního spojení pomocí parametru APN. Parametr APN určuje síť, ke které se testované zařízení připojuje. Testovaná zařízení typicky obsahují SIM karty podporující firemní APN conel.agnep.cz a APN umožňující přístup do internetu internet.t-mobile.cz. Test vlastnosti zkoumá, jestliže se změní síť mobilního připojení při změně parametru APN. Samotný test probíhá následovně. Nejdříve je zálohováno původní APN. Následuje změna APN na conel.agnep.cz. Kvůli projevení změny APN je proveden restart mobilního spojení. Dále pomocí programu mobileready je čekáno na sestavení mobilního spojení. Po sestavení mobilního spojení je zjištěna přidělená IP adresa mobilního rozhraní. Stejný postup je proveden pro zadání internetového APN. V případě, že se router v obou zadaných APN připojil do mobilní sítě, jsou porovnávány IP adresy obou připojení. Jestliže jsou IP adresy různé, tak je funkcionalita správná a test ukončen úspěšně.

9.8.4 Procedura address

Procedura address popisuje vlastnost mobilního spojení přiřazující IP adresu rozhraní mobilního spojení. Pokud známe IP adresu SIM karty v určitém APN, můžeme zadat tuto adresu jako parametr mobilního spojení. Při vytváření mobilního spojení se použije tato adresa a vytvoření spojení by se mělo provést rychleji. Test této funkcionality odpovídá zadáním IP adresy SIM karty a úspěšném vytvoření spojení. Test je prakticky implementován následovně. Nejdříve jsou zálohovány parametry routeru IP adresa a APN. Poté je z databáze zjištěna IP adresa SIM karty vložené do testovaného routeru. Po zjištění IP adresy je tato adresa zadána jako parametr mobilního spojení, druhým měněným parametrem je APN, pro které je tato adresa použita. Pro kontrolu funkčnosti zadání IP adresy je restartováno mobilní spojení a pomocí programu mobileready je čekáno na sestavení mobilního spojení. Skript je úspěšně ukončen, jestliže IP adresa SIM karty získaná z databáze je totožná s IP adresou SIM karty přidělené mobilnímu rozhraní. Skript nadále vypisuje čas v sekundách, za jak dlouho se router připojil do mobilní sítě. Na závěr skript navrátí původní nastavení routeru.

9.8.5 Procedura operator

Procedura operator popisuje vlastnost určení operátora mobilního spojení. Operátor je určen pětimístným číselným kódem. Zadáním parametru operátor je urychleno připojení do mobilní sítě, či možnost změnit operátora, pokud to SIM karta umožňuje. Testování této vlastnosti mobilního spojení probíhá následovně. V první fázi je zálohována nynější položka konfigurace operátor. Poté je z databáze zjištěn operátor SIM karty testovaného zařízení. Získaný operátor je zadán jako nový parametr testovaného zařízení operátor. Z důvodu provedení změny v konfiguraci je proveden restart mobilního spojení, a dále se čeká na nové sestavení mobilního spojení. Pokud bylo spojení úspěšně navázáno, je porovnáván přidělený operátor s operátorem zadaném jako parametr. Jestliže jsou tyto operátoři totožné, je skript ukončen s nulovým návratovým kódem. Skript vytiskne na standardní výstup dobu sestavování spojení se zadaným operátorem v konfiguraci.

9.8.6 Procedura mtu

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

ných packetů. Testování změny parametru MTU je prováděno následujícím způsobem. Nejdříve je zálohována nynější hodnota MTU a nastavíme novou hodnotu MTU na 500. Pro aplikaci změn v nastavení necháme restartovat mobilní spojení a počkáme na znovu navázání tohoto spojení. Po připojení testovaného zařízení do mobilní sítě je zjištěn název rozhraní mobilního spojení pro účely diagnostiky provozu sítě. Dále je v routeru na pozadí spuštěn ping s velikostí větší než nastavená maximální velikost packetu. Ping na pozadí je spouštěn pomocí programu z testovacího API pingb. V průběhu pingu je v testovaném routeru zároveň puštěn program pro monitorování sítě tcpdump. Pomocí tcpdumpu jsou po dobu tří sekund sledovány odchozí packety typu icmp request na mobilním rozhraní. Po příchodu logu z tcpdumpu je kontrolována velikost odesílaných packetů. Jestliže velikost packetů, je rovna maximální velikosti packetů je test úkončen s nulovým návratovým kódem, tedy byl ukončen úspěšně.

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 funkcionalita. Testovaná zařízení se k technologii EDGE připojují, pokud v místě použití není dostupná lepší technologie, či zařízení lepší technologii 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 technologii EDGE. Možnost připojení technologií EDGE je testována vynucením technologie EDGE a čekáním na navázání spojení pomocí teto technologie. Test testující 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 vynucena 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 místo

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 technologii LTE. Testování technologie LTE je prováděno totožným způsobem jako u předešlých technologií EDGE a UMTS. Jediným rozdílem v testovací proceduře 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 zařízeních, komunikujících tímto protokolem, můžeme změnit například změnou autentizace, či vytáčeného čísla. Protokol ppp používají vesměs 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 ovlivnit změnou parametru autentizace na autentizaci CHAP, poté se bude router přihlašovat do sítě jedině autentizací CHAP. Testování zvolení autentizace CHAP probíhá následujícím způsobem. Nejdříve je zálohován 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 deamona 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".

9.12.3 Procedura number

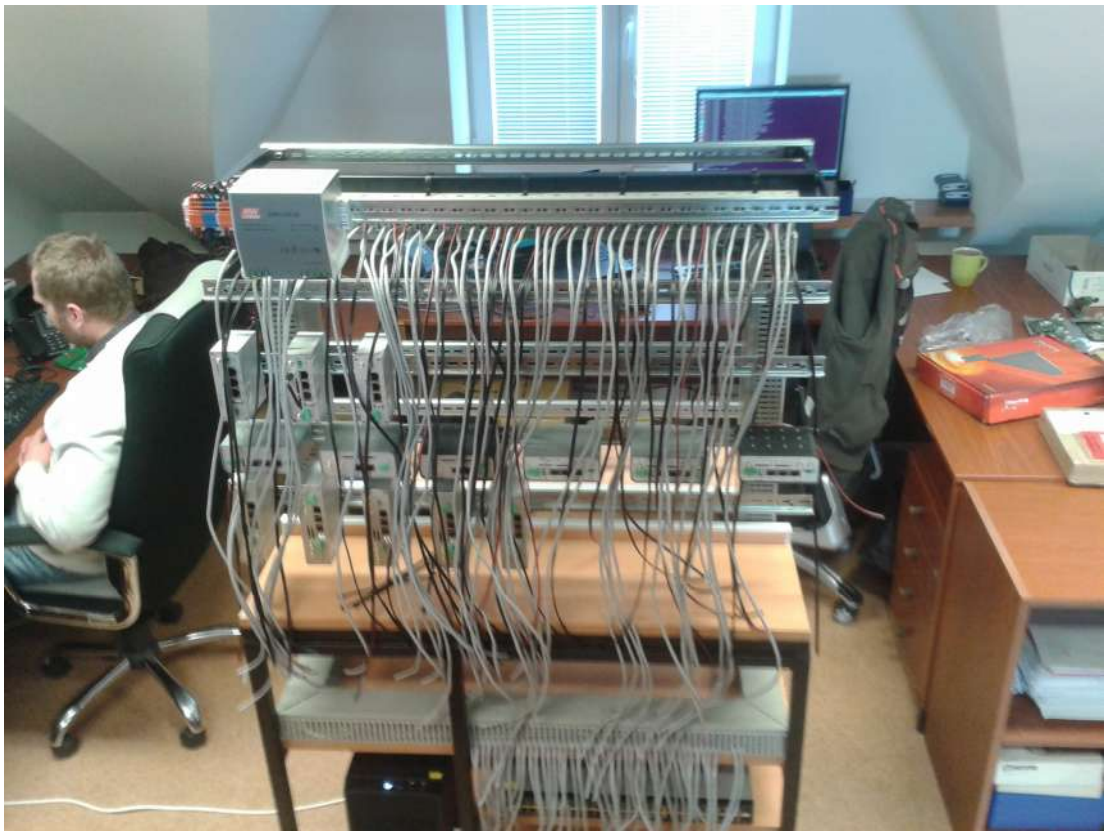
Procedura number popisuje vytáčené číslo pro PPP spojení. Defaultně vytáčené číslo zajišťuje správné spojení, avšak v odůvodněných případech můžeme toto číslo pomocí parametru number změnit. Změna čísla se projeví změnou čísla v AT příkazu ATD. Test vlastnosti probíhá následovně. Nejdříve je zálohováno původní telefonní číslo a nastaveno jakékoliv jiné číslo. Z důvodu projevení změny vytáčeného čísla je restartováno mobilní spojení. Po restartu mobilního spojení je po dobu třiceti sekund sledován systémový log testovaného zařízení a je vyhledáván AT příkaz ATD s parametrem zadaného čísla v konfiguraci. Test je prohlášen za úspěšný, jestliže je daný příkaz v logu nalezen. Nakonec je navráceno původní telefonní číslo a mobilní spojení je restartováno.

10 Testovací laboratoř v praxi

Nyní je všechna softwarová část testovací laboratoře připravena k použití a je potřeba vše začít testovat na fyzickém hardwaru. Pro osazení všech částí testovací laboratoře byl vytvořen speciální stojan. Stojan má ve spodní části poličky na různá vybavení. V poličkách je prozatím umístěn testovací server a konfigurovatelný switch. Horní část stojanu tvoří DIN lišty sloužící k umístění testovaných zařízení. Celkem je zde umístěno deset DIN lišt po dvou metrech. Pro jednodušší vedení kabeláže jsem na stojan připevnil plastové ranžírovací panely.

Obrázek 50 Neosazený stojan

Pro první fázi nasazení testování budu muset rozvést 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 ranžírovacím panelem a rovnoměrně po celé délce stojanu vyvedeny ven. Ethernetové kabely jsem vedl od switche umístěného ve střední polici stojanu do svrchního ranžírovacího panelu, kde jsou Ethernetové kabely také rovnorměrně 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 visících kabelů.



Obrázek 51 Příprava kabeláže na stojan

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 funkcionality všech instalovaných výrobků. Pomocí testovacího API lze snadno doplňovat nové testy testující další funkcionality. 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 udělán 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.

Dalším budoucím rozšířením bude testování všech dostupných rozhraní routerů. Nyní jsou testovány pouze všechny Ethernet rozhraní. V dalších fázích bude přidáno testování všech sériových rozhraní zapojením zařízení do testovacích měřičů. Pro účely testování vstupů a výstupů bude potřeba vyvinout přípravek, 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.

Správnou spolupráci hardwaru se softwarem je možné kontrolovat například měřením spotřeby každého zařízení po nahrání nového firmwaru. 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 navrhnuta 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. Měřící modul bude schopen postupně pomocí relé přivádět 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 systémem. Implementována bylo integrační a systémové testování všech vyráběných výrobků, ostatní úrovně testování by nepřinášely takový přínos u 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 skládající se základních informací o daném zařízení a všech jeho podporovaných funkcí. Podle těchto informací se na všech zařízeních pouštějí a vyhodnocují testy.

V další fázi byly zkoumány dostupné nástroje pro různé typy testování. Zde nebyl nalezen 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 integrační testování. Ve zbytku práce se tedy počítá s vývojem nového systému a použitím samostatných utilit v rámci modulů nebo API testovacího systému.

V následující kapitole byla navržena laboratoř, kde bude samotné testování probíhat. Testovací laboratoř je navržena tak, aby při samotném testování nebyl nutný jakýkoliv manuální zásah. Laboratoř si lze představit například jako LAN síť obsahující všechny testované výrobky, testovací server, konfigurovatelné switche a dále různé pomocné zařízení pro testování. Návrh sítě celé laboratoře i dalších komponentů se v průběhu vývoje nových testů ukázala jako zdařená. Všechny prozatím napsané testy fungovaly bez jakéhokoliv manuálního zásahu do testovací laboratoře.

Dále je popsán způsob implementace testování podle modelů do testovacího systému. V testovacím systému byl zvolen nejvíce abstraktní přístup k testování podle modelů. Ke každé funkcionalitě je napsána sada testovacích procedur pomocí níž je daná funkce testována. Modely jednotlivých zařízení poté skládáme právě z těchto funkcionalit a dále jejich základních vlastností. Těmito vlastnostmi může být například IP adresa zařízení nebo telefonní číslo SIM karty osazené uvnitř. Samotné testy se dále spouštějí a vyhodnocují podle těchto modelů.

Spouštění stahování zdrojových kódů, překladu, jednotlivých testů a další režii potřebnou při testování obstarává 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 snadno 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 funkcionalitu. Další funkční, nebo grafické prvky budou doplňovány dle potřeby v průběhu používání testovacího systému.

Nedílnou součásti 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 obstarávající všechny kroky, byla sestavena a testována kompletní testovací laboratoř. Během testů se neprojevily žádné závažné chyby. Drobné chyby se objevovaly v implementacích jednotlivých testů. Tyto chyby byly předpokládané a byly již odstraněny. Chyby v testech odráží skutečnost z teoretického úvodu, jenž uvádí, že pokud test je proveden chybně, tak je ve většině 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 společnosti 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

[1] Legeard B Utting M. *Practical Model-Based Testing: A Tools Approach*. Morgan-Kaufmann, 2007, s. 431. ISBN: 978-0-12-372501-1.

[2] Labiche Y. Shafique M. *A Systematic Review of Model Based Testing Tool Support*. Květ. 2010. URL: `http://squall.sce.carleton.ca/pubs/tech_report/TR_SCE-10-04.pdf`.

[3] Isermann Rolf. *Fault-Diagnosis Systems*. Springe, 2006. ISBN: 3-540-24112-4.

[4] Hanuš Petr. *Uživatelský manuál EDGE router ER75i v2*. Červ. 2013.

[5] Hanuš Petr. *Bezdrátový router SPECTRE v3 LTE uživatelský manuál*. Červ. 2013.

[6] Hanuš Petr. *Volitelný port RS232 uživatelský manuál*. Ún. 2012.

[7] Svoboda Jan Hanuš Petr. *Konfigurační manuál pro v2 routery*. Břez. 2015.

[8] Svoboda Jan Hanuš Petr. *Konfigurační manuál pro v3 routery*. Květ. 2015.

[9] Hanuš Petr. *Commands and Scripts for v2 and v3 Routers*. Led. 2015.