Expert Router Configuration

Final Report

Michael A. Perez

Spring 2015

Vision Document.....5

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

Expert Router Configuration

Vision Document

Michael A. Perez

Spring 2015

1. INTRODUCTION

PURPOSE

This document is the first in a collection of software engineering documents on the proposed software system, an Expert System for Network Router Configuration. In this Vision document we present the system at a high level of description with the intent that this should be used as a guide throughout the system development life-cycle to keep the project within scope, warding off feature creep. A discussion of the problem to be solved will be given first, leading to targeted user needs and goals. From these needs and goals, we will visually capture major features using UML Use-case diagrams. From which we will derive informal requirements or Software Product Features. Requirements will be formalized in the next volume of this collection, theSystem Requirements Specification (SRS) document.

PROBLEM DESCRIPTION

According to a published solicitation by the United States' Department of Defense Advanced Research and Projects Agency (DARPA) in 2006, incorrectly configured network devices are a common cause of insecure computer networks[1]. In the enterprise world, flawed configurations might result from fat-fingered input, poor quality network analysis, or distributed network devices with mismatched or

uncomplimentary configurations. However, in the context of a 1st world residence, typical of a small office/home office (SOHO), a single wireless router is solely responsible for implementing network configuration and policy. While an enterprise network is carefully designed using some engineering process, a SOHO network can be characterized as ad-hoc in terms of design. Most SOHO router manufacturers understand this and provide "quick setup" firmware pre-installed. The default settings on these routers are acceptable for typical SOHO use cases but the typical SOHO user is ignorant of most of these settings. And, in indeed, the default administrative password is rarely changed leaving many SOHO routers vulnerable to attack[2]. However, for the power SOHO user, there are 3rd party firmwares available, for example, Linux-based OpenWRT, DD-WRT, and Tomato. These firmwares are designed to replace a SOHO wireless router's pre-existing firmware from the manufacturer and provide new or improved features for the router that otherwise would only be available on more expensive router [3,4]. As open-source software supported by an active community, an added feature of these firmwares is support for older, inexpensive hardware and turnarounds for security fixes that are quicker than one could expect from an original manufacturer who has dropped support or has become defunct[5]. Among the three firmwares mentioned, OpenWRT is currently the only one fully open-source and free providing the lowest barrier to full access. Configuring OpenWRT, on the other hand, is a manual and knowledge intensive process requiring a domain expert in order to get the most benefit from the features provided by OpenWRT.

A solution may include an Expert System, a system that emulates the decision-making ability of a human subject matter expert (SME) by reasoning about collected facts to infer knowledge. Expert systems should not be confused with cognitive modeling programs, which attempt to simulate human mental architecture in detail. Instead, expert systems are practical programs that use heuristic strategies developed to solve specific classes of problems such the software program TurboTax by Intuit.

STATEMENT OF SCOPE

The goal of this project is a small step towards bringing down the accessibility barrier to using 3rd party firmware on consumer-grade, network routing devices, specifically OpenWRT version 10.03.1. Applications of conversational-mode expert systems such as TurboTax have proved successful at empowering the common people in doing for themselves that which they otherwise would need an expert. To achieve the stated goal, we will apply the conversation-mode of an expert system to gather information about a SOHO network from a user. The Expert system will read in production rules from a knowledge base along with the user's responses and infer a new status message to display to the user, a new question to ask of the user, and /or a new UCI command to append to a growing configuration script. After the conversational information gathering is complete, the system will allow the user to review and apply generated configuration scripts. The configuration script will be applied remotely via SSH and implemented using the Unified Configuration Interface (UCI) command-line utility developed by OpenWRT developers. At this stage of the system development, no 3rd party firmware installation help with be provided nor will the system account for every possible configuration scenario. This project will start off small by getting the user up and running with typical 3rd party firmware usage scenarios such as:

- *multi-family dwellings where the various wireless routers in close proximity may interfere with each other,*
- *a repeater network,*
- *a client-bridge network,*
- *VPN,*
- *and port forwarding.*

2. User Description

This section justifies the system's existence by examining market demographics, the user's environment, the user's goals and needs, and existing alternatives.

User/Market Demographics

Users of the system are ISP home customers and are either tech-savvy or not. The intended goal of the system is to make accessible the configuration of the OpenWRT firmware for users who 1) have never been exposed to 3rd party firmware for their SOHO routers, 2) are not familiar with Linux, and 3) are not familiar with network configurations.

User Environment

The system is meant to be used in a residential ISP network. In such a network, a single router, typically supplied by the ISP to the customer, sits between the ISP network and the customer's home network. A home network consists of laptops, desktops and related computing devices such as wireless printers and network-accessible storage. It would also include network-enabled consumer devices such as gaming consoles like the Microsoft XBOX, and network-enabled mobile devices such as smartphones and tablets. Sometimes the ISP supplied router provides wired access only and the customer may use their own personal wireless router, sitting in-line with the ISP-supplied router, to supply wireless access to the home network. This end-user supplied router is the target router to be configured by the system. Admittedly, targeting this device may not be effective at port forwarding if the ISP-supplied router has port-forwarding settings of its own. Configuration takes place rarely, such as when the user replaces the router device or when the user self-troubleshoots network connections. Currently, end-users use the factory installed web-based front-end to configure the device's port forwarding, SSID, access permissions, security, DHCP, and other network settings. This may take 5 minutes to an hour, depending on the settings to be changed, skill level of the user, and familiarity with the configuration tool.

Key User/Market Goals and Needs

GN01: Provide big payoff for configuring router

The major goal for this project to get normal users to configure security for their wireless router and minimize the number of vulnerable wireless routers on the internet. To motivate users to engage with a technological gadget they otherwise would not, a big payoff or reward must be provided. Open-source firmware such as OpenWRT can provide this payoff by providing features to users that would otherwise cost them thousands of dollars.

GN02: Bridge the Knowledge Gap

It will not be enough to just provide a big payoff, we will have to make configuration accessible to the user.

Alternatives and Competition

Formally verified configurations of networks at the Autonomous System node level within the BGP protocol space have been around over a decade[6,7,8]. However, requiring the system user to use formal logic to define their network configuration is tantamount to requiring the user be a good low-level programmer. Our project instead will be accessible to the typical residential ISP costumer.

There exist many freely available and high quality software tools to monitor traffic. However, they require the training and expertise of a network administrative professional.

Truly there is nothing on the market today that does what we aim to do here

PIKT is a complex tool used for administrator functions including automated network configuration of a single server host by copying files in place from a repository. This may complement this system well in the future but we will not focus on version control for this project.

The TCS Security Blanket product offers automated hardening, or lockdowns, of Linux and Solaris operating systems, specifically, Solaris 10 and Red Hat Enterprise Linux 5 and its derivatives. The enterprise edition allows central control over several hosts. It performs the hardening according to profiles which may be based on industry standards (e.g. DCID 6/3) typically developed by authoritative agencies or they may be custom profiles developed in-house. This product offers high-level policy specifications, verification of policy compliance, and automatic configuration. However, like PIKT, it offers these features for the host computers on the network not the router. Each device coming onto the network will need to be locked down individually and with consumers going through networked devices at least every two years this may not be desirable. Also, Security Blanket is targeted at the enterprise and not the home market. The stand-alone version would be a great substitute to this project for home users with only secure lockdowns of their networks in mind but it will put a great burden on the user to understand network, computer security terminology.

3. Product Overview

An overview of the system is given here in the context of a product perspective, market position, a summary of capabilities, assumptions and dependencies, and a preliminary project plan to include tasks/milestones, schedule, and budget.

Product Perspective

The product is a a freely available, open-sourced system for the general public. It uses the CLIPS inference engine developed by NASA at its core. The CLIPS inference engine accepts a flat file as a knowledge-base. It is important to note that this initial development of the system focuses on the OpenWRT firmware but only the knowledge-base is tied to this firmware, generating Unified Configuration Interface (UCI) commands. Other firmware, based on OpenWRT or not, may be targeted by user-defined knowledge-base flat files.

Summary of Capabilities

The product will determine a set of commands to configure OpenWRT firmware running on a SOHO router. Determination will be based solely on a user interview given by the system. User-defined knowledge-bases, in flat files, may be substituted as a video game cartridge can be swapped out of a Nintendo video game console.

Assumptions and Dependencies

The product will depend on the host platform's connection to router and the version and support for UCI of the OpenWRT firmware installed

Preliminary Development Plan

In an effort to minimize cost, the system will utilize and incorporate technology and software for which new or additional licensing is kept to a minimum. The software engineering strategy devised for this project should be driven by the methods and tools used, and the nature of the project. Methods include maintaining a single vision though out the project lifecycle, a software development plan consisting of an enumeration of tasks to be completed, a process model defining how the tasks will be completed, and a schedule defining when they will be completed. Other methods include documenting a Test plan as requirements and software are developed, developing a hierarchy of requirements from customer goals/needs to product features to software specifications. Tools used will be fairly generic and will include Gantt charts or PERT analysis, UML modeling tools, and general office productivity software. The project's high-level requirements can be grouped into standalone components indicating a modular nature of the project.

Based on these observations an incremental development process model will be used in developing a software development plan to build modules of the system and integrate them together, adding functionality as the project progresses. Certain modules provide base functionality while others provide advanced or added functionality. This will be considered when scheduling tasks. In addition, the V-model for development will be used in creating test plans as the project produces artifacts from high level to low level.

Major milestones and deliverables are listed below:

1. *Proposal (This Vision Document)*
2. *Define Acceptance tests for each Goal/Need and Product Feature.*
3. *Committee Accepts Proposal*
4. *Stable Product Features*
5. *User Manual (derived from UseCase Workflow/Alt. Flow/Exceptions)*
6. *Stable Analysis Document (UML Analysis Diagrams)*
7. *Preliminary System Design Document (UML Design Diagrams)*

a. *Design System Test Cases (black-box tests)*

8. *System Analysis Presentation to Committee*
9. *Design and Implement Each Module*
    a. *Detailed Module Design*
    b. *Update Detailed System Design*
    c. *Implement and test module for stability*

i. *Design/Implement Unit test cases (white-box tests)*

d. *Integrate with existing modules*

e. *Repeat for next module*

10. *Fully Detailed System Design Document*
11. *Final Report and Presentation*

4. KEY USE CASES

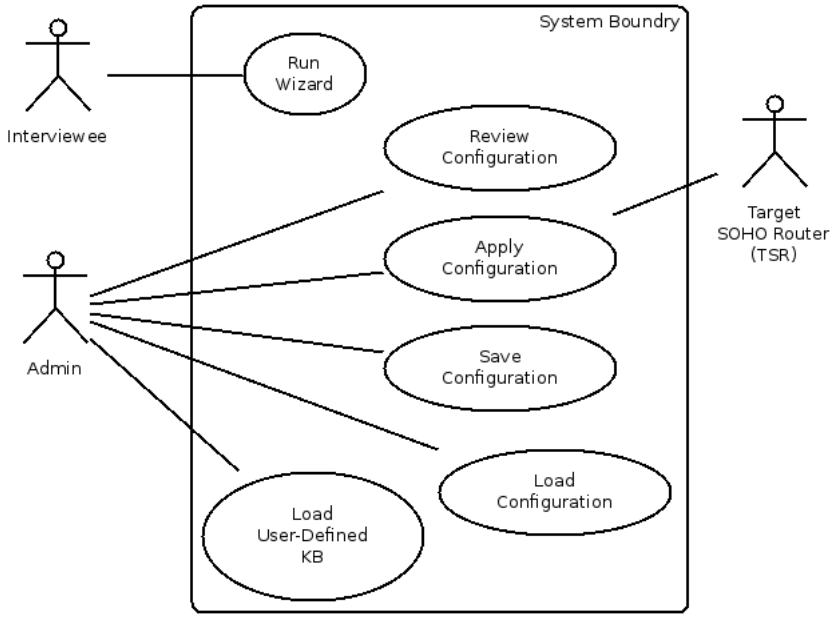

FIGURE 1: KEY USE-CASE DIAGRAM

UC01: **Run Wizard**

RUN WIZARD

Brief Description	The Wizard takes the interviewee step by step through the interview process to define the user's home network and determine how it should be configured.
Traceability	GN02, GN03
Preconditions	Wizard is not currently running. A knowledge base is selected.

Normal Flow	The user starts the wizard. The wizard prompts the user with a yes/no or multiple choice question. The user answers the question or cancels the wizard. If the wizard needs more information to determine a configuration for a SOHO router, it repeats from step 2 If the user canceled the wizard, nothing will be saved. The wizard will start from the beginning if the user starts it again
Alt. Flow	
Post Conditions	A set of shell script commands are generated to configure the target router unless the wizard was canceled.
Primary Actors	Interviewee

UC02: **Review Configuration**

REVIEW CONFIGURATION

Brief Description	Though the user is not expected to understand any of it, the set of shell script configuration commands to be run on the target router will be presented for the user's review.
Traceability	GN04
Preconditions	Wizard has completed to the end OR the user has loaded a previously saved configuration
Normal Flow	The system prompts the user with the configuration commands that will be applied to the target SOHO router. The user may review all the commands before applying them against the target SOHO router.
Alt. Flow	none
Post Conditions	The user may edit the commands, save the configuration to the file system, and apply the configuration to the target SOHO router.
Primary Actors	Admin

UC03: **Apply Configuration**

APPLY CONFIGURATION

Brief Description	Using SSH, remotely apply the configuration commands to the target SOHO router in the same LAN segment.

Traceability	GN01
Preconditions	Target router is installed with OpenWRT with SSH server enabled. System and target router are on the same LAN segment. User has SSH login credentials to the target router. A configuration file is already loaded here after the wizard has run or because the user has loaded a previously saved configuration.
Normal Flow	The user opts to apply the configuration currently being reviewed. The system prompts the user for the IP address of the target router. By default, the IP address of the network gateway is used. The user accepts the default or enters in a new IP address. The user is then prompted for the SSH username and password of the remote target SOHO router. Alternately, a public/private key pair may be used instead of a username and password credentials. Using SSH, the system remotely logs into the target router and runs the configuration script.
Alt. Flow	
Post Conditions	The SSH connection is closed. The target router is configured according to the configuration commands run by the script. The user's network connection may be interrupted.
Exception	

5. Feature Attributes

These attributes characterize all product features. Their values should be adjusted to reflect their current state as the project progresses.

Status: *Proposed, Rejected, Adopted, Implemented*

The Status attribute tracks progress during definition of the project baseline and subsequent development.

Priority: *Critical, Useful, Enhancement*

The Priority attribute ranks features by relative benefit to the end user and satisfaction of business goals and needs.

Effort: *Low, Medium, High*

The Effort attribute estimates the amount of time, lines of code, function points, or just general level of effort.

Risk: *Low, Medium, High*

The Risk attribute measures the probability that a feature will cause undesirable events such as cost overruns, schedule delays, or even cancellation.

Stability: *Low, Medium, High*

The Stability attribute measures the level of understanding of a feature.

Release: *Proposal, Plan, Module1, Module2, Module3, Module4, Module5, …, Final*

The Release attribute indicates the intended product version in which the feature will be introduced.

Assigned-To: *Person's name*

The Assigned-To attribute indicates the role or team that is responsible for further elicitation, software requirements, or implementation. Unless otherwise noted, the value for this attribute will be the author.

Traceability

The Traceability attribute tracks the source of the requested feature, e.g., one or more goals and needs from section 2 or a Use Case from section 4.

6. Software Product Features

The following software product features support the realization of one or more of the previously defined use cases.

SPF01: **Prompt user with yes/no question**

Prompt user with yes/no question

Attribute	Value	Notes
Status	Implemented	
Priority	Critical	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF02: **Prompt user with multiple choice question**

PROMPT USER WITH MULTIPLE CHOICE QUESTION

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF03: **Start wizard**

START WIZARD

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF04: **Cancel wizard**

CANCEL WIZARD

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF05: **Determine if enough information from user has been collected**

DETERMINE IF ENOUGH INFORMATION FROM USER HAS BEEN COLLECTED

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF06: **Package configuration as repeatable script**

PACKAGE CONFIGURATION AS REPEATABLE SCRIPT

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF07: **Configuration script view**

CONFIGURATION SCRIPT VIEW

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF08: **Load configuration file from disk**

LOAD CONFIGURATION FILE FROM DISK

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF09: **Configuration script editor**

CONFIGURATION SCRIPT EDITOR.

Attribute	Value	Notes
Status	Proposed	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	Low	
Release		
Assigned-To		
Traceability	UC01	

SPF10: **Save edits to configuration script**

SAVE EDITS TO CONFIGURATION SCRIPT.

Attribute	Value	Notes
Status	Proposed	
Priority	Useful	
Effort	High	
Risk	High	
Stability	Low	
Release		
Assigned-To		
Traceability	UC01	

SPF11: **Remotely apply configuration script via SSH to the target SOHO router (TSR) that resides in the local LAN segment.**

REMOTELY APPLY CONFIGURATION SCRIPT VIA SSH.

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF12: **Determine gateway IP address of current LAN segment**

DETERMINE GATEWAY IP ADDRESS OF CURRENT LAN SEGMENT.

Attribute	Value	Notes
Status	Proposed	
Priority	Useful	
Effort	Low	
Risk	Medium	
Stability	High	
Release		
Assigned-To		
Traceability	UC01	

SPF13: **Select IP address of target router**

SELECT IP ADDRESS OF TARGET ROUTER.

Attribute	Value	Notes
Status	Proposed	
Priority	Useful	
Effort	Medium	
Risk	High	
Stability	Low	
Release		
Assigned-To		
Traceability	UC01	

SPF14: **SSH login with username/password credentials**

SSH LOGIN WITH USERNAME/PASSWORD CREDENTIALS.

Attribute	Value	Notes
Status	Adopted	
Priority	Critical	
Effort	High	
Risk	Medium	
Stability	Medium	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF15: **SSH login with public/private key pair credentials**

SSH LOGIN WITH PUBLIC/PRIVATE KEY PAIR CREDENTIALS.

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF16: **Generate Public/Private key pair**

GENERATE PUBLIC/PRIVATE KEY PAIR.

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF17: **Select Knowledge-base to use for wizard**

SELECT KNOWLEDGE-BASE TO USE FOR WIZARD.

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

SPF18: **Load default knowledge-base**

LOAD DEFAULT KNOWLEDGE-BASE.

Attribute	Value	Notes
Status	Implemented	
Priority	Useful	
Effort	High	
Risk	Low	
Stability	High	
Release	Final	
Assigned-To		
Traceability	UC01	

7. OTHER PRODUCT REQUIREMENTS

Non-functional requirements and their priorities are described here at a high-level. Applicable standards, hardware, or platform requirements; performance requirements; and environmental requirements. The quality ranges for performance, robustness, fault tolerance, usability, and similar characteristics that are not captured in the Feature Set are defined here. If useful, attributes such as priority, stability, effort, and risk are described.

APPLICABLE STANDARDS

Whenever possible, encryption will be used when sending or receiving configuration data from the network. Consideration for a high Shannon Entropy will guide selection for keys, passwords, and/or algorithms. Follow procedures to ensure configurations are applied only by authorized users.

CONSTRAINTS AND DEPENDENCIES

The system requires TCP/IP network compatible devices.

There is some method available to remotely configure the router.

The system is cross-platform, able to run on Microsoft Windows, Mac OS X, and Linux with X Windows.

PERFORMANCE REQUIREMENTS

The system shall have a peak load of 1 active users.

The system shall have a maximum response time of 60 seconds for target router ping query before timing out.

DOCUMENTATION REQUIREMENTS

Define any specific documentation requirements, including user manuals, online help, installation, labeling, and packaging requirements.

User Manual

As defined in section 1, a key feature of this system is the accessibility to novice users. Therefore, a user manual will be provided. The Normal and Alternative flows from the Use Cases will guide the creation of the user manual.

Installation Guide

A guide will be written in which the installation will outlined.

Labeling and Packaging

The system is developed and packaged as a single unit. Any ancillary systems (i.e. database, server software, etc.) are not provided.

Licensing Installation

N/A

Expert Router Configuration

Software Requirements Specification

Michael A. Perez

Spring 2015

1. INTRODUCTION

PURPOSE

This document presents formal requirements required for project development. Specifications for these requirements are documented here with enough detail to define system behavior. All requirements defined here can be traced back to targeted user needs and goals in the Vision document. This document overlaps greatly with both the Analysis and the Design phases and as such may contain UML diagrams from both. UML diagrams will be marked with their source phase.

OVERVIEW

This document classifies requirements as one of functional, non-functional, or other. Functional requirements describe the behavior of the system and the software architecture. Non-functional requirements describe the characteristics of the system, such as design constraints, and describe the system's technical architecture. Classifying requirements as "Other" is a catch-all method to describe requirements that are neither functional nor non-functional. Document requirements fall under the "Other" classification.

WRSPM REFERENCE MODEL

A reference model for using formal methods in developing requirements was described by [4]. Formal methods are mathematical techniques, Church's higher-order logic in this case, used by software

engineers to solve problems in system verification and specification. This reference model defines the relationship between a system and its environment by the interaction of five artifacts: World, Requirements, Program, Machine, and Specifications. Simply, this WRSPM reference model states that requirements must be logically satisfied by the phenomena of the World artifact in which the system exists and the Specifications artifact which define the interface between the environment and the system. The World artifact defines all phenomena of the environment while the Machine artifact defines all phenomena of the system. The Requirements and Program artifacts specify the phenomena which are desirable of the environment and system, respectively. Using Church's higher-order logic, [4], state that W ^ S => R, or that the World and Specification artifacts logically satisfy the Requirement artifacts. Note that in the figure below the (S)pecification artifact is the interface between the Environment and the System.

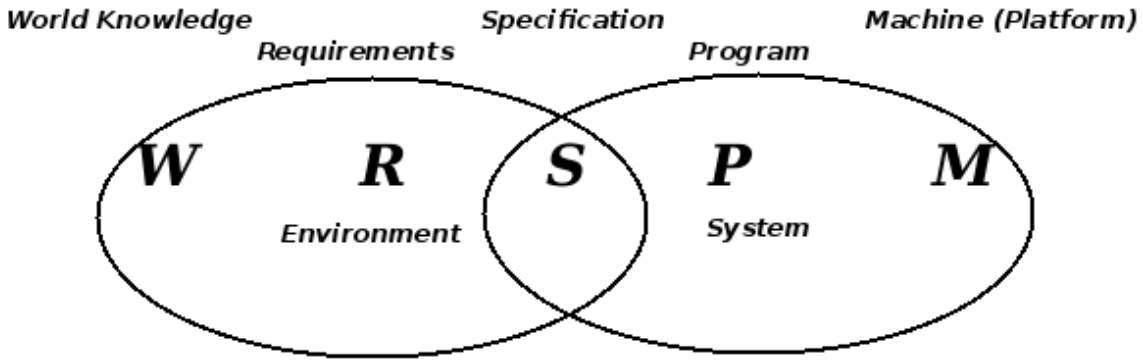

FIGURE 2: WRSPM REFERENCE MODEL

The phenomena of the World and Machine artifacts are known, non-functional requirements so we will discuss them in *section 2 Non-Functional Requirements*. Functional requirements are a product of the Requirement artifact, our the Software Product Features of the Vision document, and will be more formally discussed here in *section 3 Functional Requirements*. Phenomena of the the Specification artifact give our functional requirements verification and validity, so they, too, will be included in *section 3 Functional Requirements*. Phenomena of the Specification artifact will be derived from W, M, and R of our WRSPM model and from UML analysis diagrams as needed. The Program artifact is the software to be developed, an unknown at this point, and, as such, is not discussed in this document.

2. USE-CASE REALIZATIONS

To uncover any hidden or missed requirements from what we have already discussed in the Vision document, we will use UML analysis diagrams to realize our Use-cases.

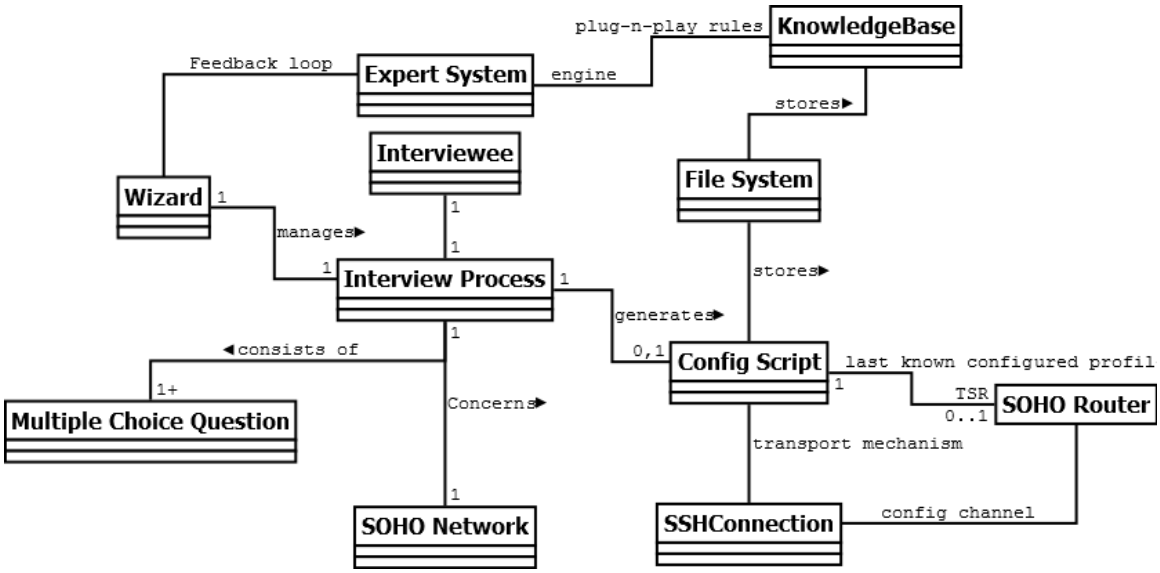

FIGURE 3: UML ANALYSIS CLASS DIAGRAM

3. NON-FUNCTIONAL REQUIREMENTS

This section defines non-functional requirements which specify overall system characteristics such as cost and reliability. Non-functional requirements drive the technical architecture of a system.

WORLD (OR DOMAIN KNOWLEDGE) (**W** - *R* - *S* - *P* - *M*) "W restricts the actions that the environment can perform"
W1. The system user is not knowledgeable in network configurations.
W2. The system user can operate a computer Graphical User Interface (GUI)
W3. The system user will consider any response time over 7 seconds to be an indication of unresponsiveness.
W4. The system user has the SSH credentials of the targeted SOHO router (TSR)
W5. One TSR exists for each system user.
W6. The TSR is visible to the system user while operating the system.

TABLE 1: WORLD ARTIFACTS OF THE WRSPM REFERENCE MODEL

MACHINE (PROGRAMMING PLATFORM) (*W - R - S - P -* **M**) "M restricts the actions that the system can perform"
M1. All pertinent networks use the TCP/IPv4 network stack.
M2. The operating system may be Microsoft Windows, Mac OS X, or Linux.
M3. A single targeted SOHO router (TSR) provides wireless and wired network connectivity between an ISP gateway router and a residential ISP customer's network.
M4. All pertinent networked devices use the TSR as their network gateway
M5. The TSR is running OpenWRT version Backfire 10.03.1 firmware
M6. The firmware is running an SSH server using SSH protocol version 2
M7. The firmware supports network configuration through the command line utility known as the Unified Configuration Interface (UCI)
M8. It uses the CLIPS inference engine developed by NASA
M9. The CLIPS inference engine accepts a flat file as a knowledge-base.
M10. The knowledge-base is TSR firmware specific
M11. The CLIPS inference engine is TSR firmware agnostic
M12. Configuration script data will be stored in flat files.
M13. Public/private key pairs for SSH login are stored using the RSA key format as a pair of files on the local file system.
M14. The system runs an SSH client using SSH protocol version 2
M15. The system and the TSR communicate using SSH tunnels only

TABLE 2: MACHINE ARTIFACTS OF THE WRSPM REFERENCE MODEL

4. FUNCTIONAL REQUIREMENTS

This section defines the capabilities and functions that the System must be able to perform successfully. Functional requirements drive the application architecture of a system. Our functional requirements come from the Software Product Features from the vision document.

REQUIREMENTS (*W -* **R** *- S - P - M*) "R says which of all possible actions in W are desired"	
Traceability	Requirement

SPF01	R1. The system shall prompt user with Yes/No question
SPF02	R2. The system shall prompt user with Multiple Choice question
SPF03	R3. The system shall start the wizard when the user says so
SPF04	R4. The system shall cancel the wizard when the user says so
SPF05	R5. The system shall determine if enough information from user has been collected
SPF06	R6. The system shall package configuration into a repeatable script
SPF07	R7. The system shall provide a configuration script viewer
SPF08	R8. The system shall load a configuration file from disk
SPF09	R9. The system shall provide a configuration script editor
SPF10	R10. The system shall save configuration commands to a file that can later be executable on the target SOHO router (TSR)
SPF11	R11. The system shall remotely apply a configuration script via SSH to the target SOHO router (TSR) that resides in the local LAN segment.
SPF12	R12. The system shall determine the gateway IP address of the local LAN segment
SPF13	R13. The system shall determine the IP address of the target SOHO router (TSR)
SPF14	R14. The system shall login to the TSR via SSH with username/password credentials
SPF15	R15. The system shall login to the TSR via SSH with public/private key pair credentials
SPF16	R16. The system shall generate public/private key pairs for use in SSH logins
SPF17	R17. The system shall load a knowledge-base to use for the wizard
SPF18	R18. The system shall load a default knowledge-base when a user-defined knowledge-base is not provided

TABLE 3: REQUIREMENT ARTIFACTS OF THE WRSPM REFERENCE MODEL

We now define Specification artifacts and, because of Adequacy as defined previously (W ^ S => R), we will use these specifications as our method of requirement verification and validity.

SPECIFICATIONS (*W* - *R* - **S** - *P* - *M*) "S properly takes W into account in saying what is needed to obtain R"	
Traceability	**Specification to verify and validate the referenced requirements**
R1, R2, R5	S1. The inference engine uses initial facts from the knowledge-base and user responses gathered up to this point to infer the next set of actions to take. S2. When the user answers a question, the inference engine matches the user's response and existing assertions with the antecedents of unactivated rules. S3. The knowledge-base stores each user question as part of the consequent of a rule S4. The knowledge-base stores UCI configuration commands as part of the consequent of a rule S5. When an activated rule's consequent includes a UCI command, the command is appended to the configuration script. S6. When an activated rule's consequent includes a user question, the user is prompted with the question. S7. When the list of activated rules' consequents do not include a user question, the wizard should complete the resulting configuration script should be saved as a flat file to disk.
R6, R7, R8, R9, R10	S8. The configuration script is a flat file consisting of a collection of UCI commands, and other commands as needed, that are executable on the target SOHO router (TSR) running OpenWRT. S9. The system saves the script file on the local filesystem after running the wizard through to completion S10. The system loads the script file into a view for the user to review and edit after the user loads a previously saved configuration script. S11. The script file view allows simple text editing of the configuration script file. S12. The script file view allows saving the configuration script to a flat, text file using a filename of the user's choice.

R11-16	S13. The system shall run an SSH client to connect to the SSH server running on the target SOHO router (TSR) S14. The system will prompt the user for the SSH username and password of the target SOHO router (TSR) each time it connects and will not store the credentials in persistent data storage on the system. S15. The system is able to generate its own private/public key pairs for use with SSH logins and distribute the public key to the target SOHO router (TSR) S16. The system will treat default network gateway IP address as the target SOHO router (TSR)
R3-4	S17. When the wizard is not running, a button should be marked in the system's GUI such that when clicked the wizard starts. S18. When the wizard is actively running, a button should be marked in the system's GUI such that when clicked the wizard is interrupted and stops running.
R17-18	S19. When the wizard is not running, the user is able to select a knowledge-base flat file from local disk for the wizard to use the next time it runs. S20. If the user starts the wizard without choosing a knowledge-base file, a default knowledge-base file should be available to use.

TABLE 4: SPECIFICATION ARTIFACTS OF THE WRSPM REFERENCE MODEL

5. OTHER REQUIREMENTS

DOCUMENTATION REQUIREMENTS

USER MANUAL

A key feature of this system is the accessibility to novice users. Therefore, a user manual is an important part of the project deliverables. The Normal and Alternative flows from the Use Cases will guide the creation of the user manual.

INSTALLATION GUIDE

A guide will be written in which the installation will outlined.

LABELING AND PACKAGING

The system is developed and packaged as a single unit. Any ancillary systems (i.e. database, server software, etc.) are not provided.

LICENSING INSTALLATION

N/A

Expert Router Configuration

Software Design Descriptions

Michael A. Perez

Spring 2015

1. Introduction

1.1 Purpose

This document's audience is is intended to be those who will implement and verify the correct functioning of the system. Descriptions of all software components required for successful project development and specifications for these components are documented here with enough detail to be implemented by component developers. All software components defined here can be traced back to formal requirements in the SRS document.

1.2 Design Overview

This document addresses the system's software architecture, that is, the identification and organization of all software components of the system. An Object-oriented (OO) approach is taken in this document.

2. System Overview

The project will implement an Expert System using a user-defined knowledge base and an inference engine to produce the configuration of a small home/small office computer network. The system will aditionally apply the configuration script against the router running OpenWRT 10.03.1.

3. System Architecture

3.1 Architectural Design

The Software Architecture/Subsystem diagram is presented here. Subsystems provide an interface when they implement software routines within their area of responsibility needed by other components. And they require an interface when they need to make calls to software routines outside of their area of responsibility.

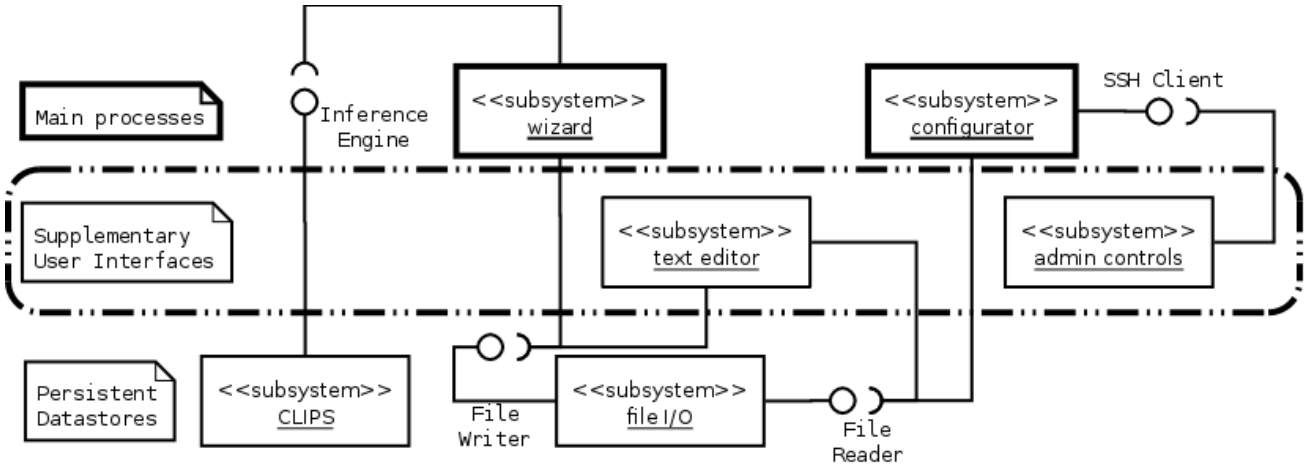

Figure 4: System Architecture

From the system architecture we see that there are three categories of subsystems: Main Processes, Supplementary User Interfaces, and Persistent data stores. A full component diagram is presented after we decompose each subsystem into classes.

This architecture uses an Expert System and an SSH client for its main processing duties. An SSH Client can be used to manage SSH credentials such as a public/private key pair and to remotely execute configuration scripts on the targeted SOHO router (TSR). In this case, the TSR should be running OpenWRT and the dropbear SSH server. An Expert System can be used to interview the user regarding the SOHO network to be configured. CLIPS6.24.1 is a freely available public domain Expert System developed by NASA in the late 1980's and early 1990's. As of this writing, it is actively maintained by the public on sourceforge.net. Its basic user interface is command-line-based and its command syntax resembles the LISP programming language. A graphical user interface is available for CLIPS in Microsoft Windows and Apple Mac OS X. It also runs successfully on Linux under Wine.

3.2 Decomposition Description

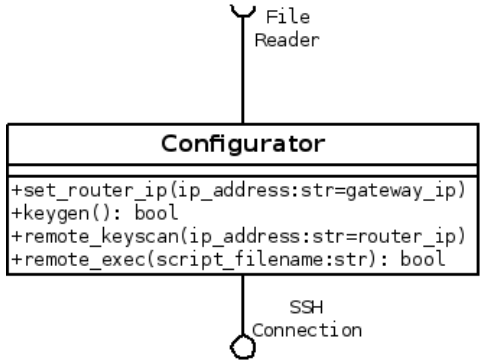

FIGURE 5: CONFIGURATOR SUBSYSTEM: CLASS DIAGRAM

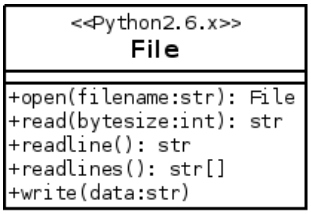

FIGURE 6: FILE IO SUBSYSTEM: CLASS DIAGRAM

The configuration and File I/O subsystems are pretty straight-forward and consist on one class each. The CLIPS subsytem on the other hand consists of three main classes. The python library, pyclips, allows integration of our system with an isolated CLIPS process managed by the pyclips library. With pyclips our system can programmatically interact with CLIPS to handle control, errors, and input/output.

The CLIPS subsystem and our Wizard subsystem will communicate using an Interviewer interface. This interface will provide our Wizard subsystem with a means to gather information from the user in a conversational format and allow the CLIPS expert system to reason towards a network router configuration based on the collected data. . The wizard subsystem heavily relies on pyclips and the Pyramid web framework. The DeclarativeBase class provides an Object-to-Relational Database Mapper (ORM) so any classes based on that will have instant access an API to manage data in the database. The ORM implementation is called SQLAlchemy. The ViewCallable class is really a decorator for any python function or class that will handle incoming web requests. The ScopedSession class provides our system with database transaction management. Extending this session with the ZopeTransactionExtension allows our transactions to be mapped one-to-one with each web request. This means that every request-response cycle will result in a rollback or commit in the database of any data modifications during that cycle.

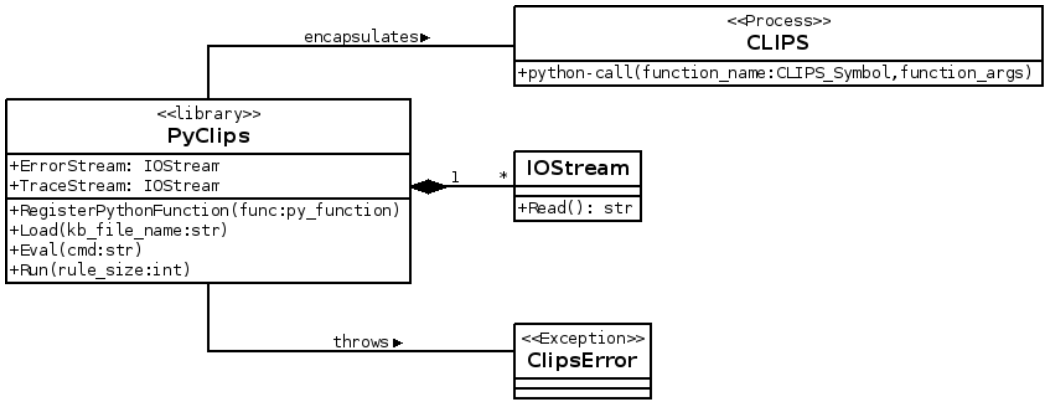

FIGURE 7: CLIPS SUBSYSTEM: CLASS DIAGRAM

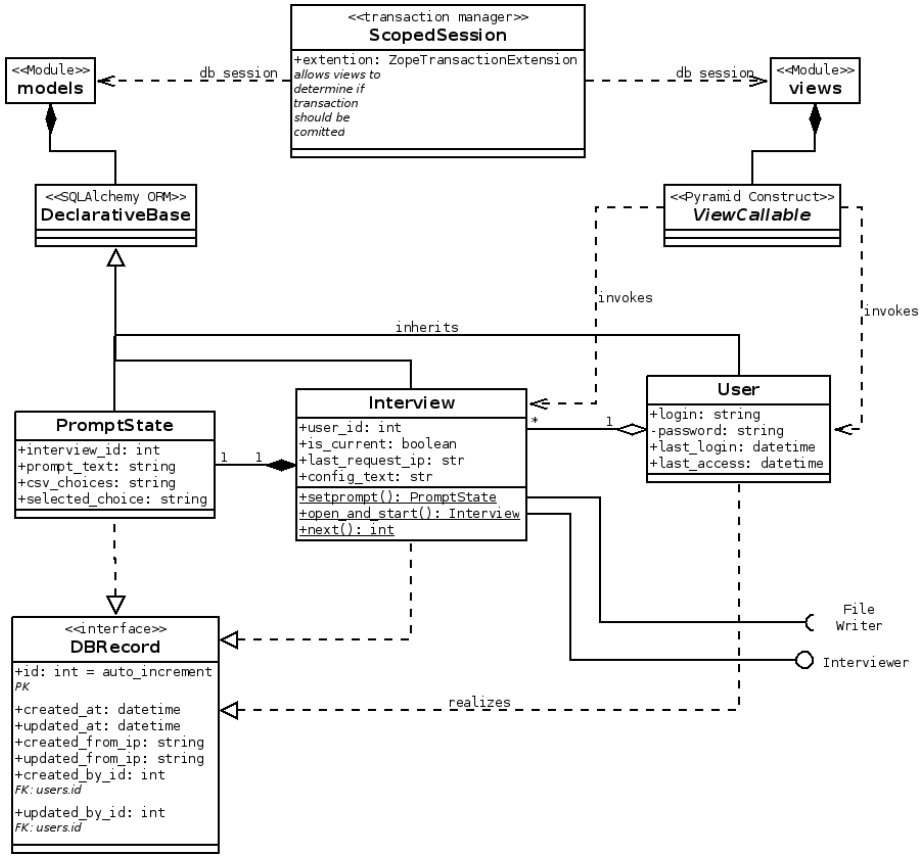

FIGURE 8: WIZARD SUBSYSTEM: CLASS DIAGRAM

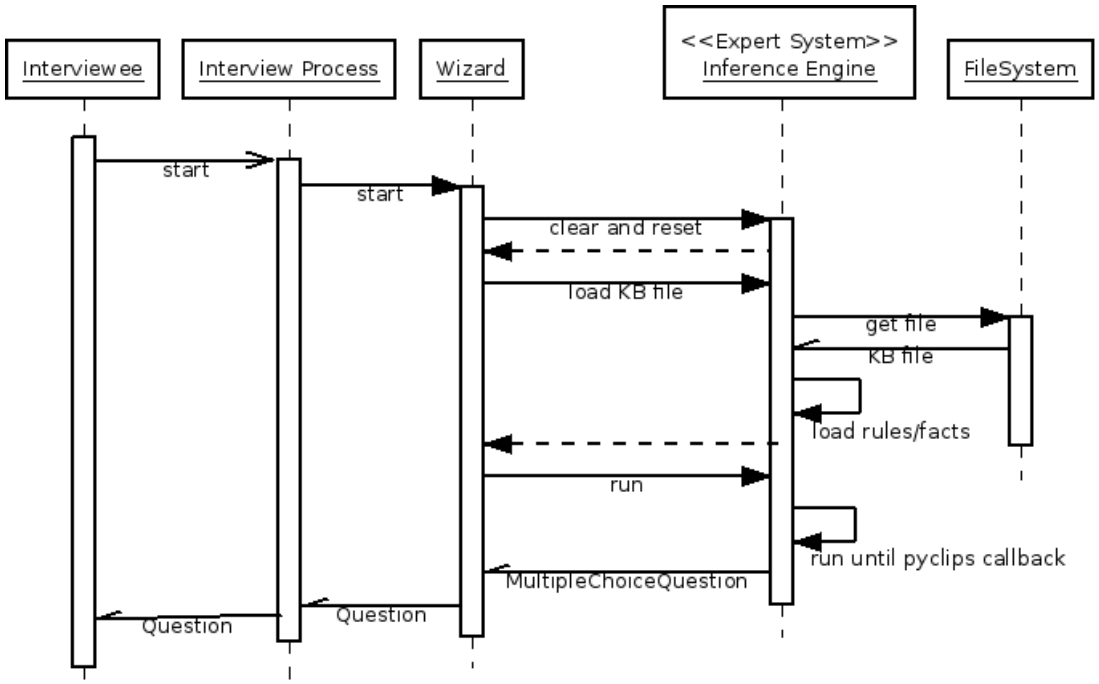

FIGURE 9: START WIZARD: SEQUENCE DIAGRAM

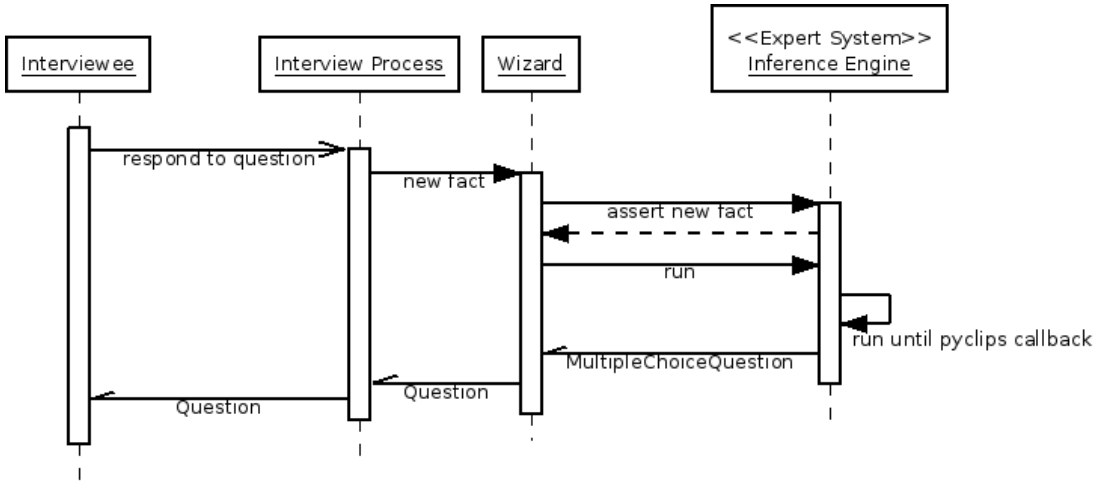

FIGURE 10: ONGOING INTERVIEW: SEQUENCE DIAGRAM

We have grouped these classes into six main components of the system and have deployed our components in the deployment diagram below.

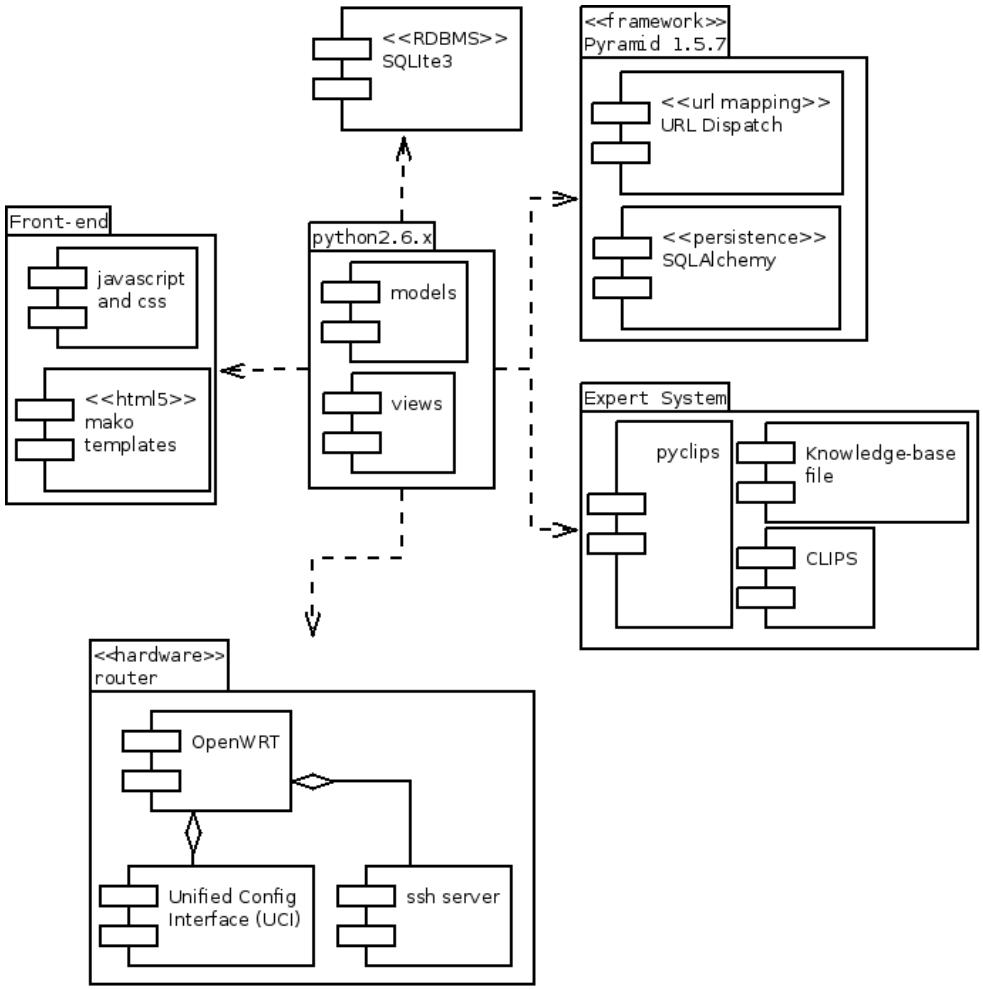

FIGURE 11: COMPONENT DIAGRAM

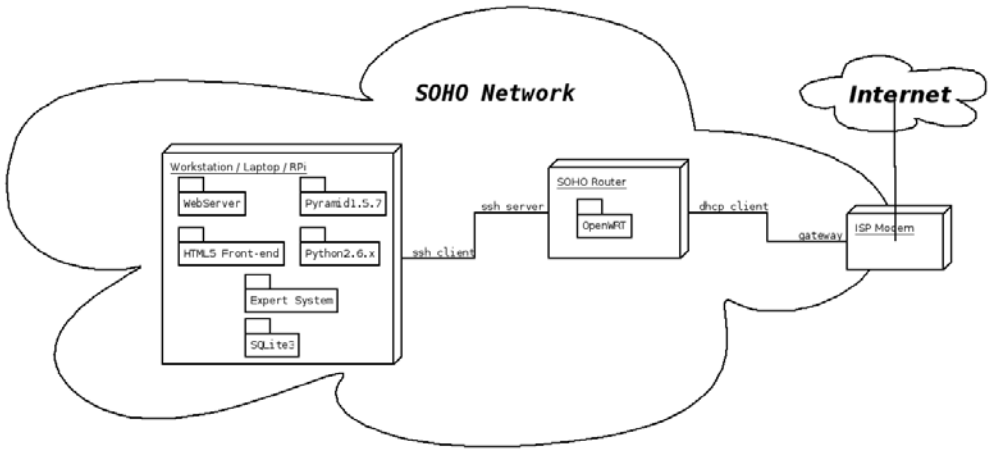

FIGURE 12: DEPLOYMENT DIAGRAM

3.3 Design Rationale

An alternative to our chosen architecture is to forego the Expert System and use a collection of IF-THEN-ELSE or SWITCH-CASE statements. The advantage of the Expert System in this case is its ability to efficiently handle a large knowledge base. Specifically, Expert Systems handle the problem known as Combinatorial Explosion with pattern-matching algorithms such as Rete. Consider a system with n variables. Each variable as Z possible values. This system then has Z^n possible end-states. A system with 4 variables and just 2 possible outcomes per variable has 4^2=16 possible outcomes. Adding just one more variable increases the number of outcomes to 5^2=25. The high number of variables involved in developing a network configuration and the exponential nature of the Combinatorial Explosion problem means that adding one new rule to an already high number can be a major maintenance headache. So instead of trying to hard-code every possible combination with an IF-THEN-ELSE or SWITCH-CASE, we instead can rely on the pattern-matching ability of an Expert System and the knowledge representation syntax of its knowledge-base. On the other hand, expert systems are not meant to handle systems with few variables since a lot of effort goes into knowledge engineering. This comes to the main disadvantage of Expert Systems, the quality of the knowledge gathered. The Knowledge-base itself cannot configure a router. This system will need to integrate with the Expert System as it runs the interview process.

The Wizard component provides a simple, easy to use graphical user interface (GUI) to input computer network usage information. It should conform to good design heuristics such as

1) Visibility of system status where clear instructions are presented and feedback for any action is featured prominently,
2) a Match between the system and the real world by using the vocabulary of the user in place of the technical vocabulary of the network configuration and the system,
3) Flexibility of use is satisfied by allowing the user to skip input prompts which he or she cannot answer, and
4) Aesthetic and minimalist design where each input prompt presents only information needed to respond to the input prompt.

The CLIPS subsystem is a conversational-mode Expert System including the knowledge-base written . Expanding on the aesthetic and minimalist design heuristic first discussed in section 4.1 Wizard, the Wizard interview questions provided by the knowledge base do not force users to come up with an answer. Instead the knowledge base gives the user an out by including an "I do not know", "Skip this question", or similar response choice in the multiple choice questions sent along with the question itself to the Wizard subsystem.

3.3.1 Knowledge-base Design

Several approaches to knowledge base design are known such as the use of certainty factors, backward-chaining, and decision trees. When using certainty factors, the "expert advice" given by the system can be shown to have a quantitative confidence level. For example, it may calculate all factors used to derive the end solution and state that the system has a confidence level of some percentage on the solution presented. This is useful for systems only providing advise to the user who will ultimately make the final decision. However, the goal of our system is determine a configuration script for OpenWRT in a transparent and useful way for the user who knows nothing about OpenWRT configuration. The use of

backward-chaining presumes that an ideal or otherwise optimal goal is known ahead of time. Such a design approach is one that tries to determine if the various criteria for the goal are met. A decision-tree, or forward-chaining, is an approach that works towards a goal by gathering facts and inferring the implications of those facts. In the context of propositional logic, this is a logical argument form known as modus ponens. For example, if P implies Q, and P is asserted, then Q must be true as well. Backward-chaining uses the same principle but infers the assertions from the implied goal. Forward-chaining gathers data from the user and works toward finding any solution, optimal or not.

Forward-chaining would be adequate for a small decision tree knowledge-base but we anticipate this knowledge base to grow and to be maintained across a rotating queue of domain experts. Backward-chaining would allow a much larger knowledge-base without having the user to traverse the much larger tree. With backward-chaining the user would only need to traverse some path much closer to the optimal one. CLIPS supports only forward-chaining but we can simulate backward-chaining using rules.

4. Data Design

4.1 Data Description

In order to derive the data points needed for the system, we should examine the desired behavior as described in the statement of scope in our Vision document.

"To achieve the stated goal, we will apply the conversational-mode of an expert system to gather information about a SOHO network from a user. The Expert System will read in production rules from a knowledge base along with the user's responses and infer a new status message to display to the user, a new question to ask of the user, and/or a new UCI command to append to a growing configuration script. After the conversational information gathering is complete, the system will allow the user to review, edit, save, and apply the generated configuration file. The configuration script will be applied remotely via SSH."

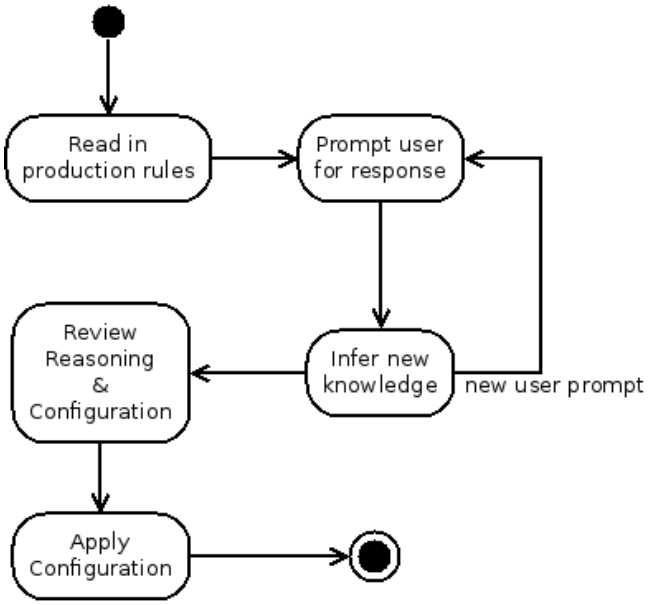

Figure 13: Statement of Scope: Activity Diagram

The above Activity diagram models this statement of scope at a high level. In the following sections we will present details at a lower level to decompose the system into two major activities: the Wizard and the Configuration.

4.1.1 Wizard Activity

The Wizard subsystem carries out an interview-style process using a graphical user interface and a conversational-mode Expert System. An Expert System can be described as a three-part process called Match-Resolve-Act (MRA) carried out by an inference engine. The MRA process runs through the following steps described by Asheeh Goja [6]:

1) The inference engine uses a pattern matching algorithm to create or update an **agenda** of production rules whose antecedents are satisfied with the asserted facts in **working memory**. The rules are made up of two groups of patterns: the antecedent, or Left-hand side (LHS), and a consequent, or the Right-hand side (RHS).

2) A **conflict resolver** selects the rule with the highest priority from the agenda to trigger.

3) An **action executor** executes the consequent of the selected rule and also removes this rule from the agenda. 4) If there are rules leftover in the agenda, repeat from step 1.

This cycle continues until there are no rules left in the agenda. In our application of Expert Systems, after a rule is triggered it must never be matched or triggered again until the system is reset. This is an important feature of Expert Systems called refraction which prevents rules from firing repeatedly in the Match-Resolve-Act cycle [7].

The interview process builds on the MRA sub-process by prompting the user for dynamic feedback, asserting the facts from the user's response and fed back into the MRA sub-process. The outputs from the MRA sub-process are a new interview query or status message to display to the user and/or a new shell script command to append to the growing configuration file.

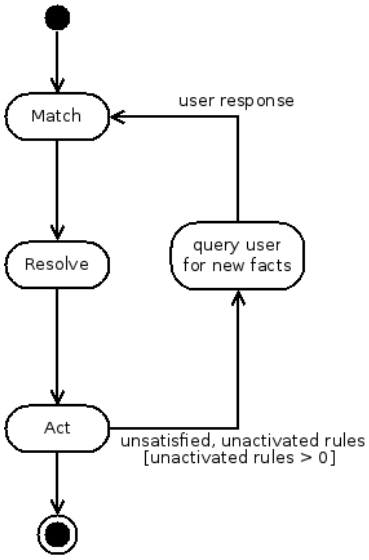

Figure 14: Wizard: Activity Diagram

CLIPS will handle the dynamic data such as the Agenda and our system should be responsible for keeping track of the interviews conducted and the configuration script origins and reasoning. The rules and initial facts should be stored for access outside of the interview process. Flat file storage will allow portability and human-readability of the knowledge-base. It is vital to this system that there be no barriers for the domain-experts in the public to pick up on this project and contribute to the knowledge-base.

4.1.2 Configuration Subsystem

The Configuration subsystem is simpler than the Wizard subsystem. It takes one of three commands from the user to load, save, or apply a UCI configuration file. To load or save a file, only requires a filename parameter. But to apply a configuration file using SSH requires the IP address of the targeted SOHO router (TSR) as well. Additionally, an SSH username and password will be needed if a private/public key pair has not been established between the system and the TSR. When saving or loading a file, the system should also display the status of the file operation to the user using the output display. The status presented should indicate if the operation is ongoing or complete. The same goes for the status of the SSH operation when applying the configuration. If an error occurs over SSH, the return error code or message should be displayed.

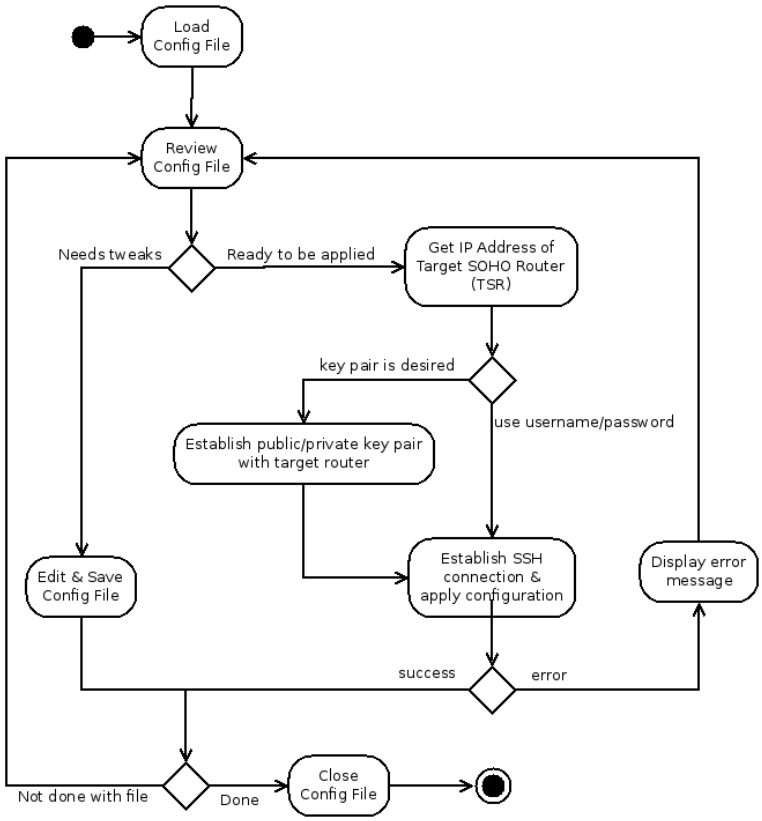

Figure 15: Configuration: Activity Diagram

Expert Router Configuration

Software Test Plan

Michael A. Perez

Spring 2015

1. Introduction

Purpose

This document defines the overall plan to validate functional requirements specified in the Software Requirements Specification (SRS) document. Test cases will also be defined here, detailing the steps needed to validate each functional requirement.

Overview

This document classifies requirements as one of functional, non-functional, or other. Functional requirements describe the behavior of the system and the software architecture. Non-functional requirements describe the characteristics of the system, such as design constraints, and describe the system's technical architecture. Classifying requirements as "Other" is a catch-all method to describe requirements that are neither functional or non-functional. Document requirements fall under the "Other" classification. The test plan will consider all functional requirements for testing and those non-functional requirements that place constraints on the system.

2. Testing Strategy

The test cases developed here will be based on an analytical approach to the system requirements, especially on those requirements that are functional or that specify design-constraints. Traceability across use cases, requirements, and software product features provide various levels of detail as appropriate for each test case. For example, preconditions and steps for a test case can be based on the Use Case, the Requirement, or the Product Feature.

A model-based approach would also be appropriate here. A predefined list of responses to the Expert System's line of questioning should produce the queries in the same order and result in the same UCI configuration script each time.

3. Features to be Tested

Because of the level of technical specifics used in the language of the Specifications developed in the SRS, they will require low-level unit testing for validation. Because the Specifications developed in the SRS are written with implementation details in mind, they require low-level unit testing and are not appropriate here to validate the goals of the project. Here we will concern ourselves with the validation of the higher level Functional Requirements.

4. Test Cases

Name	TC01
Brief Description	The system should allow the user to start the Wizard and to stop or cancel the Wizard anytime during the interview process so that the user is not stuck in the interview process when they cannot answer any more questions.
Traceability	R3, R4
Preconditions	System is running Wizard is not initiated Home screen is presented on screen

Steps	Initiate the configuration Wizard Cancel and Reset the Wizard Enter the following initial facts in the first screen. Submit the initial facts form. Cancel and Reset the Wizard. Enter the initial facts again as in Step 3. Submit the initial facts form again. Respond Yes or No to the next prompt. Cancel and Reset the Wizard
Expected Results	The wizard's interview process is canceled and re-started without issue in Steps 2, 5, and 9

Name	TC02
Brief Description	The system should be able to prompt the user for certain types of responses. This test case will validate that the system can present the user with a predefined list of responses and accept one item from that list as a response.
Traceability	R1, R2
Preconditions	Test case, TC01 passes and was immediately performed before Initial Wizard facts form is presented on screen
Steps	Submit the following initial facts: Respond Yes to the prompt "XYZ" Cancel and Reset the Wizard Resubmit the initial facts from Step 1. This time respond No to the prompt from step 2.
Expected Results	The wizard should present the same prompt for the same initial facts in steps 2 and 5. Responding Yes to this prompt should result in X while responding No to this prompt should result in Y.

Name	TC03
Brief Description	The interview process should end in a reasonable amount of time.
Traceability	R5, R6, R7
Preconditions	Test case, TC02 passes and was immediately performed before Initial Wizard facts form is presented on screen
Steps	Submit the following initial facts: From now on out, respond with the following in the order shown: Yes, No Yes, Yes, No Review the UCI script
Expected Results	After the last response in Step 2, the Wizard will have enough information and will stop to produce a UCI configuration script. In step 3, the System will present the UCI configuration script on screen.

Expert Router Configuration

Results and Conclusion

Michael A. Perez

Spring 2015

1. Conclusion

Results

The system is able to produce a simple configuration script based on a short interview with the user. And the interview questions can be extended by adding new python modules and new rules to the Knowledge-base.

Discussion

Of the top five common issues in software projects, two definitely plagued this project. Insufficient end-user involvement and an inaccurate estimate of needed resources.

The lack of native support for backward-chaining in the knowledge base, CLIPS, presented some challenges. For the small scale of rules in this project, however, the simulation of backward-chaining was effective. However, this simulation has not been tested on a large-scale knowledge base. A major problem in this project was acquiring Subject Matter Experts (SME) to add facts to the knowledge-base. The knowledge-base was developed from OpenWRT How-Tos and Recipes but it is unclear how well those How-Tos include practical knowledge or heuristics[10]. In the meantime, a never-ending cycle of research ate all development time. Too much time was spent on researching networking scenarios and ways to maintain a knowledge base. Time should have been spent integrating CLIPS and the web framework.

This has been a learning experience on what is manageable for one individual and the author would like to thank his committee for encouraging him to down-size from his original, unmanageable, proposal.

Throughout this project there has been a struggle to reconcile the theoretical with the practical. At one point, the system architecture was developed by using Data Flow diagrams. UML diagrams were presented afterwards but these Object-oriented artifacts were incompatible with the Data Flow artifacts which were derived from functional decomposition. Software engineering document templates from IEEE were closely followed for the Vision and SRS documents but many parts of the Software Description Document template had to be re-worked.

The WRSPM reference model was a big risk for this project since it is very new concept. However, never has developing requirements been easier. With the WRSPM reference model, the World and Machine entities inform and, more importantly, provide constraints on requirement and specification development.

2. Future Work

This is just a first step in enabling users a better experience with their routers. The quality of the knowledge base will directly reflect the quality of this software product. However, knowledge representation is hard and indeed, the main disadvantage of Expert Systems is quality of the Knowledge-base. One approach to alleviate the difficulty developing the knowledge-base is Business Domain Development (BDD). Software testing packages such as Cucumber, Lettuce, and Behave, allow non-technical domain experts, or subject matter experts (SME), write test cases for software. This allows the SMEs to define software test cases in natural language text. BDD then is analogous to writing a unique scenario for a fixed software engine to carry out, just like a knowledge-base is written for an inference engine. BDD should then be considered to allow SMEs to develop knowledge-bases as natural language text files.

3. Source Code

Source code versioning for this project was managed with a local git repository. The source code may also be browsed publicly online at github.com at http://github.com/mkpz/erc

4. References

The following list of articles and journal papers are referenced throughout this report provided here for the reader's convenience.

[Fahmy 1996] Hany I. Fahmy and Christos Douligeris. "**Automatic Network Modeling, Simulation, and Performance Evaluation**" The 1996 Southcon Conference. Orlando, FL. Pp 516-520. 1996

[Tamboise 2003] Guillaume Tamboise. "**HOW-TO SECURELY USE SNMP ON A BGP/MPLS VPN NETWORK.**" http://www.sans.org/reading_room/whitepapers/networkdevs/howto_securely_use_snmp_on_a_bgp/mpls_vpn_network_245. SANS Reading Room, SANS Institute. Oct. 31, 2003.

[STTR 2006] "**AUTOMATED WIDE-AREA NETWORK CONFIGURATION FROM HIGH-LEVEL SPECIFICATIONS.**" Dept. of Defense (DoD) Small Business Technology Transfer (STTR). Agency: DARPA. Topic: Information Systems. STTR Solicitation 2006 (closed April 2006) http://www.dodsbir.net/SITIS/archives_display_topic.asp?Bookmark=28970. Retrieved May 17 2009.

[Purviance 2013] "Don't use Linksys Routers." http://superevr.com/blog/2013/dont-use-linksys-routers/. Posted by Phil Purviance on superevr.com – The Exploitation Vulnerability Research Blog on April 5, 2013. Retrieved April 6, 2013.

[Zetter 2009] "**TIME WARNER CABLE EXPOSES 65,000 CUSTOMER ROUTERS TO REMOTE HACKS.**" http://www.wired.com/threatlevel/2009/10/time-warner-cable/ Posted by Kim Zetter on wired.com on October 20 2009. Retrieved April 1, 2013.

[Chiappetta 2012] "**HOW TO ENHANCE YOUR ROUTER WITH OPEN SOURCE FIRMWARE.**" http://www.pcworld.com/article/260281/how_to_enhance_your_router_with_open_source_firmware.html Posted by Marco Chiappetta on pcworld.com on Aug 3, 2012. Retrieved April 6, 2013.

[Pash 2006] "**HACK ATTACK: TURN YOUR $60 ROUTER INTO A $600 ROUTER.**" http://lifehacker.com/178132/hack-attack-turn-your-60-router-into-a-600-router Posted by Adam Pash on lifehacker.com on June 6, 2006. Retrieved April 1, 2013.

[Barrett 2009] Matthew Barrett, et. al. "**GUIDE TO ADOPTING AND USING THE SECURITY CONTENT AUTOMATION PROTOCOL (SCAP) (DRAFT).**" NIST Special Publication SP-800-117. http://csrc.nist.gov/publications/PubsSPs.htmevernote.l#SP-800-117. May 5, 2009

[Gunter 2000] Gunter, et. al. "**A REFERENCE MODEL FOR REQUIREMENTS AND SPECIFICATIONS.**" IEEE Software. May/June 2000. http://www.research.att.com/people/Zave_Pamela/library/publications/refmod.pdf§ Retrieved April 2013

[Feamster and Balakrishnan 2003] Nick Feamster and Hari Balakrishnan. "**TOWARDS A LOGIC FOR WIDE-AREA INTERNET ROUTING.**" In ACM SIGCOMM Workshops Future Directions in Network Architecture. Karlrugh, Germany. August 2003

[Feamster 2003] Nick Feamster. "**PRACTICAL VERIFICATION TECHNIQUES FOR WIDE-AREA ROUTING.**" ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-II), Cambridge, MA. November 2003.

[Feamster and Balakrishnan 2004] Nick Feamster and Hari Balakrishnan. "**VERIFYING CORRECTNESS OF WIDE-AREA INTERNET ROUTING.**" Proc. 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI), San Francisco, CA. March 2004

[Fredlund 2007] Lars-Ake Fredlund and Hans Svensson. "**MCERLANG: A MODEL CHECKER FOR A DISTRIBUTED FUNCTIONAL PROGRAMMING LANGUAGE.**" ICFP'07, October 1–3, 2007, Freiburg, Germany.

[Ouimet 2008] Martin Ouimet. "**Formal Software Verification: Model Checking and Theorem Proving**." Embedded Systems Laboratory Technical Report (ESL-TIK-00214) Massachusetts Institute of Technology(MIT). Cambridge, MA. 2005-2008

[Narain 2004] Sanjai Narain. "**Network Configuration Management via Model Finding**." Telcordia Technologies, Inc. Piscataway, NJ. 2004

[Wells 2008] Quentin Wells. "**Guide to Digital Home Technology Integration: Residential integration series.**" ISBN 1435100623, 9781435400627 Delmar Cengage Learning, Clifton, NY. 2008

[10] http://wiki.openwrt.org/doc/howto/start