Department of Computer Science

COS 301 - Software Engineering

Not Like This
Functional Requirements

Authors:	*Student number:*
Jedd Shneier	13133064
Duncan Smallwood	13027205
Daniel King	13307607
Muller Potgieter	12003672

May 27, 2016

Software Requirements Specification and Technology Neutral Process Design

Network Visualizations interface for large scale networks/Main Project

Version: Version 1.0 Beta For further references see gitHub. May 27, 2016

Contents

1 Introduction

2 Vision

3 Background

4 Architecture Requirements

4.1 Access Channel Requirements

Only registered members of Amazon's EC2 will be allowed to use the visualizer, as it will be integrated into EC2. Clients will be able to use any device(s) that can make use of Chrome, Firefox and/or Internet explorer. The browsers will not need any additional plugins to use the visualizer.

4.2 Quality Requirements

1. The system should show the following performance characteristics:
    - It should cater for large and small customers. (The customer's size is in terms of the number of networks[VPCs] and network interfaces)
    - The visualization should render all displays in under 10 seconds.
    - The visualization should render all displays in under 10 seconds.
    - The site should load and allow customer interaction within 3 seconds on a local network.
    - The project should work within normal EC2 API throttling limits.
    - The web page component should require no more computing resources than can be provided without noticeable slow down on a low end consumer laptop.
2. The system should use secure authentication supported by the EC2 API's and follow best practices

4.3 Integration Requirements

4.4 Architecture Constraints

5 Software Architecture

5.1 Architectural patterns or styles

Layered Architecture

The layered architecture separates various logical layers, specifically in this system the Presentation Layer, The Network Analyser, and The Network. The Network Analyser layer will interact with the Network layer and produce an abstraction of various data with regards to the Network. This is then passed to the Presentation Layer which will create a visual representation of the abstraction produced by the lower level.

Reasons use layered architecture:

- Flexibility, maintainability and scalability. Scalability being a major factor in the overall success of our system.
- It will also allow for teams to work on different parts of the application in parallel, thus speeding up the implentation process.
- It also makes it possible to configure different levels of security to different levels. As some levels require more security than others namely any level dealing with EC2 user's account information.
- This also allows for components to be tested independently of each other.

Some potential issues:

- A major issue with this pattern is that it is possible that it causes a performance decrease as there is an overhead of passing through layers. However, it is still untested and may be nonexistent or negligable.

5.2 Architectural tactics or strategies

Thread pooling

In thread pooling m threads work on n tasks. Usually m is less than n thus less memory is used for these threads. This will be especially helpful in this system as it will improve scalability which will be needed for the very large networks we will have to work with.

It can also improve performance as the program no longer has the overhead of creating a new thread for each new task and the system is less likely to have a bottleneck due to high memory usage.

Caching

Due to the system needing to constantly update caching is going to be extremely useful in increasing the performance of the system, as data can be continiously cached and only updated as time progresses rather than completely replaced with some of the data having not changed at all.

5.3 Use of reference architectures and frameworks

5.4 Access and integration channels

5.5 Technologies

6 Functional requirements and Application Design

This section will be completed in later stages of development

6.1 Use case prioritization

6.2 Use case/Services contracts

6.3 Required functionality

6.4 Process specifications

6.5 Domain Model

7 Open issues

- The client has yet to mention specific details on how they intend to expand on the base project.