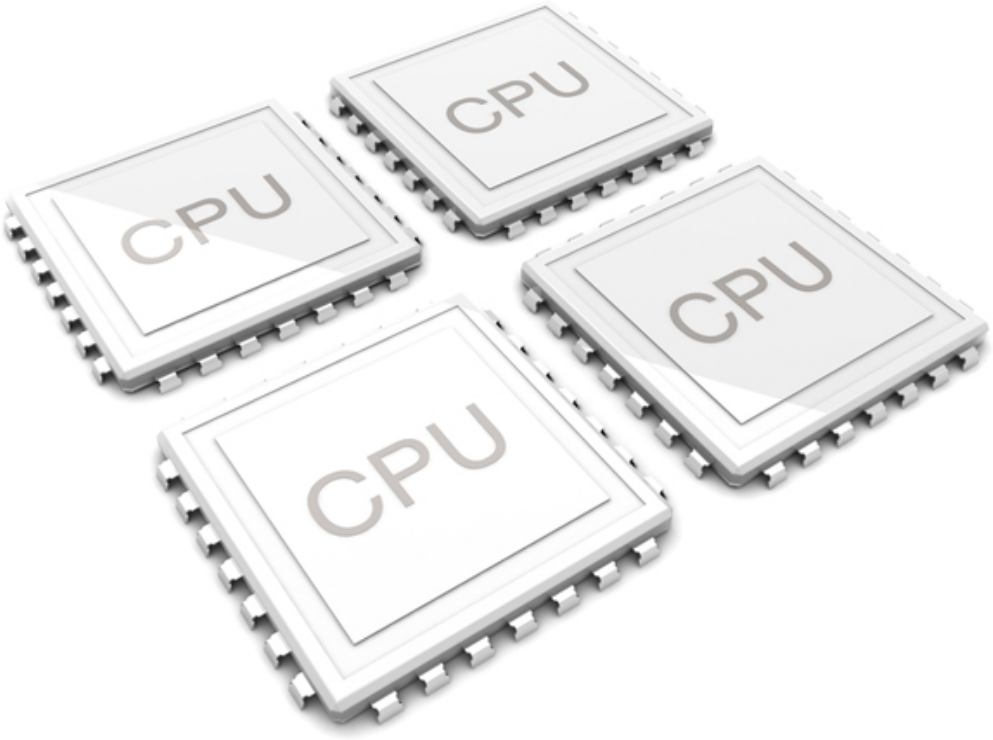
Architectural Requirements

Project: Traffic Camera Image Analysis

Client: DPSS, CSIR

Team: Quadcore Productions

Author(s):	*Student number(s):*
Mpho Baloyi	14133670
Hlengekile Jita	14077893
Mayimela Moses	14019702
Mbhele Themba	14007950



University of Pretoria, Department of Computer Science

29 July 2016

Contents

Table 1: Version History

Version No.	Date	Changes	By
1	27 May 2016		Mpho Baloyi, Hlengekile Jita, Moses Mayimela, Themba Mbhele
2	29 July 2016		Mpho Baloyi, Hlengekile Jita, Moses Mayimela, Themba Mbhele

1 Introduction

This document describes the architectural requirements of the Traffic Camera Image Analysis System. The target system will run on a web server and be accessed by users on an Android Application which will provide the users with the necessary functionality to access real-time traffic information that assists them with things such as avoiding traffic and choosing the best alternative routes.

In this specification we will cover the architectural scope of the system, the quality requirements, integration and access channel requirements and architectural constraints. In addition we will describe the software architecture we will use by describing the architectural patterns and tactics we will use as well as which frameworks and technologies will be used to achieve them.

2 Vision

For this project we aim to achieve a system that makes use of images obtained from highway cameras to provide users with up-to-date real-time traffic information. The system should simplify the user's travels by providing traffic information and notifying them before they depart of traffic conditions, calculating arrival times based on traffic conditions and help them select the most suitable route for their trips using the traffic information and additional metrics. Our vision for the target system is that it should be reliable and perform relatively quickly, both for user satisfaction and in order to be the an up-to-date traffic information system.

3 Background

As a commuter, traffic is something that is a part of everyday life, and it is not one of the more pleasurable aspects of life. Already there is software in place that assists us in dealing with this problem, such as Google Maps. This software uses the crowd-sourcing of GPS data in order to provide their up-to-date traffic information.

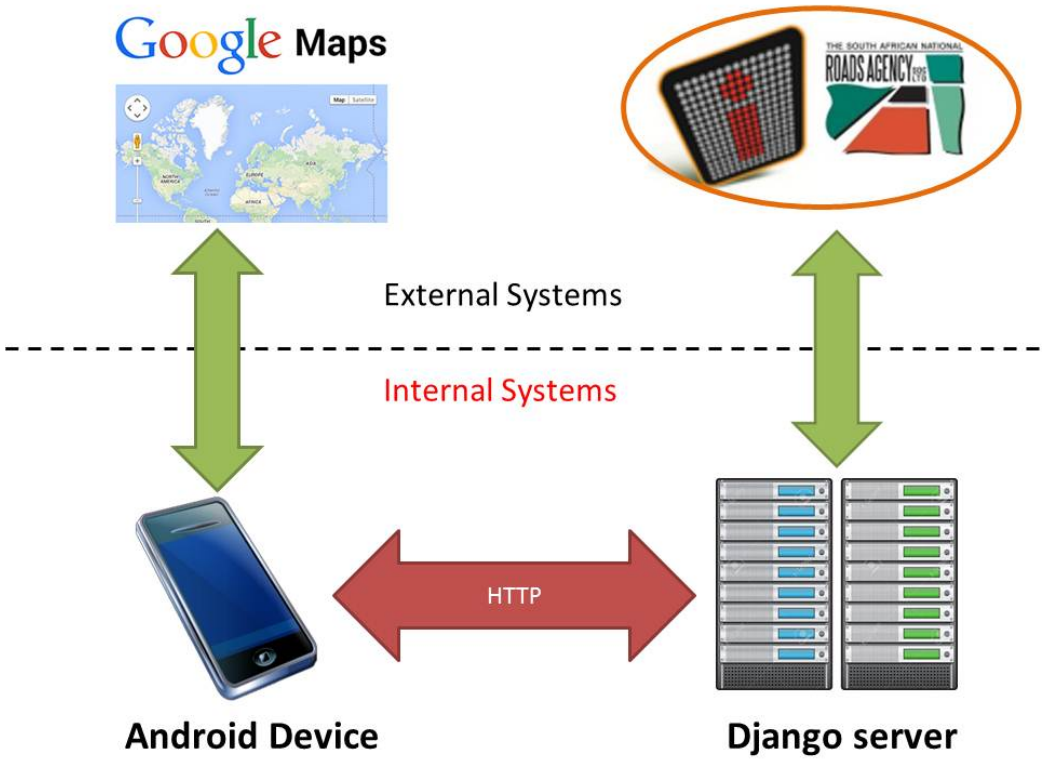
In our system,we want to take an image analysis approach to solve the same problem. We will make use of the publicly available SANRAL highway cameras to get images which can be found on https://www.i-traffic.co.za/traffic/cameras.aspx. Processing these images we will perform image analysis and determine the traffic conditions in the area of a camera. Using this information we will be able to generate information pertaining to user specified routes in order to

provide the information necessary to help them avoid traffic and choose the most suitable routes in order to do so.

4 Architectural Requirements

4.1 Architectural Scope

Below is a basic representation of the complete system. The image shows how the systems communicate and how. The image also depicts which systems are internal and which are external.



Basic representation of the complete system.

The primary responsibility of the system is the gathering of traffic images provided by the various cameras that are situated at a number of locations by the courtesy of SANRAL through their web interface, in order to perform the necessary analysis on each image. The analysis phase of the system is concerned with determining how much traffic one would encounter if using a route where traffic images are available so that a report may be generated and returned to a client. Another responsibility of the system, which aims to generate reports in a timely manner, is the persistance of the results that are

obtained through the analysis process. By persisting the results, the analysis phase does not need to be done when it is not necessary to do so. This will greatly increase the performance and efficiency of the system.

4.2 Quality Requirements

4.2.1 Usability

Usability is the key component of the system because user satisfaction is a high priority for this system because it aims to alleviate a problem that individuals experience daily. The aim is to create that is easy to understand and thus, easy to learn how to us.

How to achieve usability

- The interface of the mobile application will be designed in an intuitive manner meaning that it will be easy to navigate and easy to understand.It will include buttons and other methods of interaction for which their functionality is self explanatory.
- The interface will only have what is required and nothing more to ensure that it is not cluttered and does not cause confusion.
- Usability tests will be conducted to ensure the successful implementation of a user friendly application.

4.2.2 Scalability

Scalability refers to a systems ability to handle a heavy workload. We have identified this as an important aspect of the system because a report will have to be provided at a very fast speed because traffic changes rapidly and we need to keep up with the traffic updates. Also, multiple users will be making use of the system at the same time and they all need to be catered to at the same levels of performance.

How to achieve usability

- To improve the performance of the system, reports will only be generated when it is necessary to do so. To elaborate on this, the website from which we will be getting the images from updates its images at one minute intervals so between that period, we will not attempt to grab new images but will make use of the images that were grabbed at the previous interval. In this way, no unnecessary processing takes place.

- We will not generate a new report if it is not necessary to do so. If we generate a report for one user and another user requests a report for the exact same route when it is not time to update the images, we will use the same report that was generated for the first user. We will do this by implementing a cache server to store route reports.

4.2.3 Maintainability

The components of the system need to implemented in a manner that makes it easy for them to be updated or replaced.
How to achieve maintainability

- Maintainability will be achieved by writing loosely coupled code.

4.2.4 Reliability

Reliability is of utmost importance because a lot of people will be using the application, especially at traffic peak hours thus the system has to get the job done and it should do it effectively so that users get the best service possible.
How to achieve reliability

- The main focus will be placed on the prevention of faults. This will be done by testing the system extensively and working towards fixing any fault that is encountered.

4.3 Integration and Access Channel Requirements

The system is heavily integrated with the services provided by Google. More specifically, it is integrated with the services that are provided by the Google Maps API, Google Maps Places API, and the Google Maps Directions API. The protocol that is used by the system to integrate these services is a https request that is made to a server that is provided by Google.
Although the system makes use of a lot of services provided by Google, it is not dependent on them.
The system is implemented in a manner such that a different service provider can be plugged in, in the event that the Google services are no longer are availabe. The system is decoupled.

The system is available to clients in the form of a mobile application. The mobile application allows clients to set up routes for which they would like to receive traffic updates on. Once these routes have been set up, the client

is able to switch between the routes and a traffic report will be generated on the selected route. The report that is generated is then conveyed to the client in both a graphical and a textual manner.

4.3.1 Communication between Android device and Google maps

The communication between Google maps servers and the Android device makes us of an HTTP request from the device, requesting directions from the server. The server then responds with JSON-formatted response to the Android device. The response from Google maps has directions that must be taken by the user from the starting point to the ending point. These directions are in a form of lat-long pairs to guide the user. These lat-long pairs are sent to the server for response with camera information regarding whether or not a point is covered by the SANRAL cameras.

4.3.2 Communication between the Android device and the Django server

This communication is also initiated by the the Android device. In this case the Android device will be requesting information about the cameras in which a point falls in the range of lat-long pairs of each camera. The Android device will run a GET request to the server with the relevant lat-long information then the server will respond to that request with a JSON array which has all the cameras that matched that point.

4.3.3 Communication between Django server and itraffic website

SANRAL provides the images and also an API which has information about each camera on the road such as the camera's latitude and longitude coordinates. This information will be very useful when users request information from the server. The itraffic website currently has 644 camera feeds which are updated every minute. The script executed from the server will pull these images from the itraffic website, perform analysis and persist the results of the usable images.On the server side, the images will be analyzed and linked to their physical location on the map to help determine which cameras correspond to a request made by the user.

This communication is initiated by the server machine. The server creates http requests to the itraffic website then pull the images from the 65 pages of camera coverage. After pulling the images they are processed and their

traffic level is persisted directly to the database. The Django server will request camera information from the itraffic website with a GET request then the itraffic website responds in XML-formatted text or JSON-formatted text depending on what was specified when the request to the server was made.

4.4 Architectural Constraints

4.4.1 Network connectivity

One of the major constraints is network speed, if the android device does not have a healthy connection to the network this will make the mobile application to be slow due to waiting for responses from the external systems that it has to communicate with.

4.4.2 itraffic website

The API does enforce a limit of one request per minute, this is more evidence for the data to be persisted on a database. At the time of writing, it takes four minutes to fetch all 644 images from the SANRAL website. Thus, the one minute limitation will not impact the operation of the server or mobile application.

4.4.3 Technologies

- Python: Python will be used to implement the web server of the system which will perform the necessary processing of data and after completion, the server will send the data to the mobile application via Google Cloud Messaging.
- OpenCV: OpenCV is an image processing library that the server will make use of to process the traffic images from SANRAL.
- Android Studio: Android Studio will be used to design a mobile application on which traffic reports will be displayed.

5 Initial Software Architecture Design