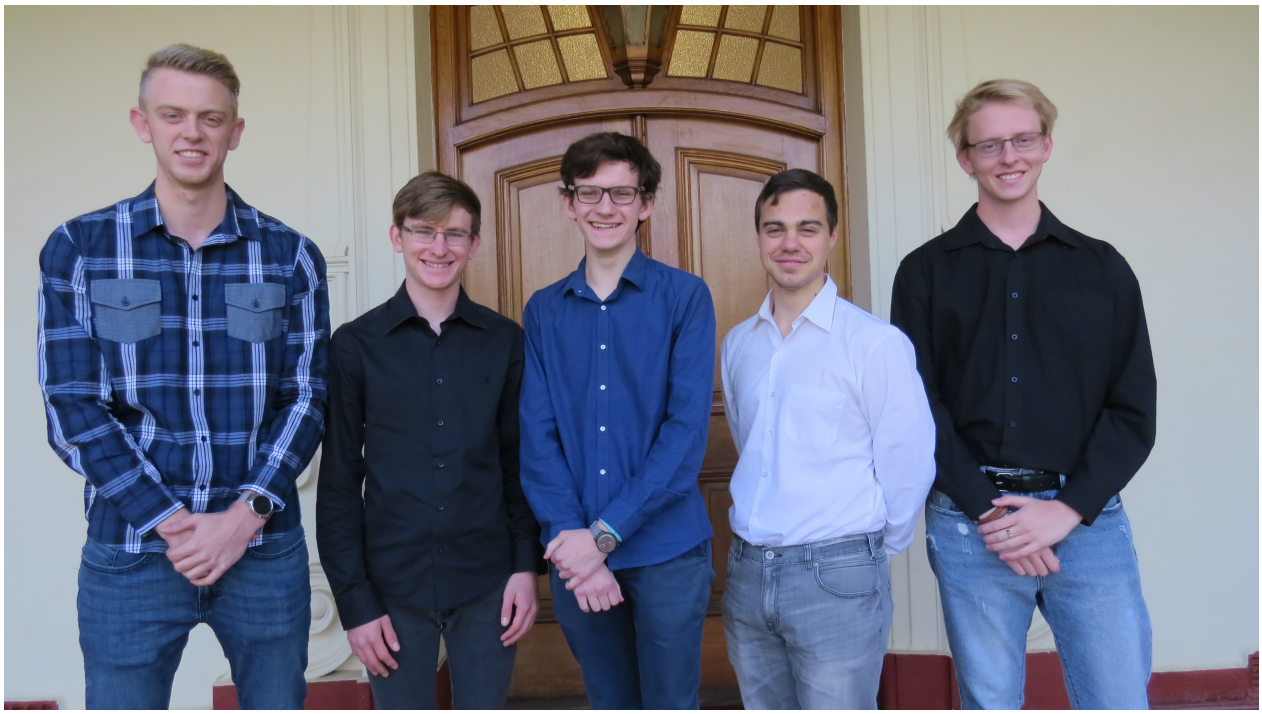
UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Architectural Design Specification

Team: Java the Hutts

2017



Nicolai van Niekerk
nicvaniek@gmail.com

Marno Hermann
marno@barnton-consulting.co.za

Stephan Nell
nellstephanj@gmail.com

Jan-Justin van Tonder
J.vanTonder@tuks.co.za

Andreas Nel
nel.andreas1@gmail.com

Contents

1 Introduction

1.1 Overview

This document identifies the architectural design specifications that satisfy the functional requirements proposed in the System Requirements Specification. It addresses the needs of the various subsystems' non-functional requirements focusing on the quality attributes, architectural patterns as well as constraints and integration requirements of the Hutts-Verification system.

The following modules' designs are included in this document:

- Image Processing
- Extraction
- Verification

The document begins with an explanation on the chosen architectural pattern and how it will be used to modularize the API. It then outlines the architectural design of each module along with all relevant UML diagrams. Finally the chosen technologies are discussed as well as the deployment of the system.

1.2 Architectural Pattern

The Hutts-Validation system will be designed as a WebAPI using the Service-Oriented Architecture (SOA). In this pattern, the architecture is essentially a collection of independent services that communicate with each other. Each service is well-defined, self-contained and does not depend on the context or state of other services, ensuring loose coupling. The API will consist of three independent services: Image Processing, Extraction and Verification.

2 Deployment Diagram

The following Deployment diagram depicts the architecture of the system as deployment of artifacts to deployment targets.. This can be seen as an overall view on the system in a deployment scenario.

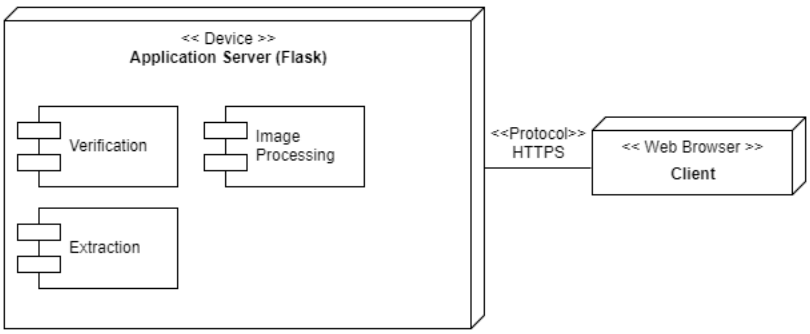


Figure 1: Deployment Diagram

3 Services

This section outlines the design of each individual service and in terms of UML diagrams and how services are related to one another.

3.1 Image Processing

TODO: Some description about the service

3.1.1 Class Diagram

The class diagram of the Image Processing module makes use of a combination of two well-known design patterns:

- **Decorator:** Different preprocessing techniques are added to the PreProcessor object at runtime.
- **Strategy:** Each preprocessing technique's *preProcess()* algorithm can be changed at runtime.

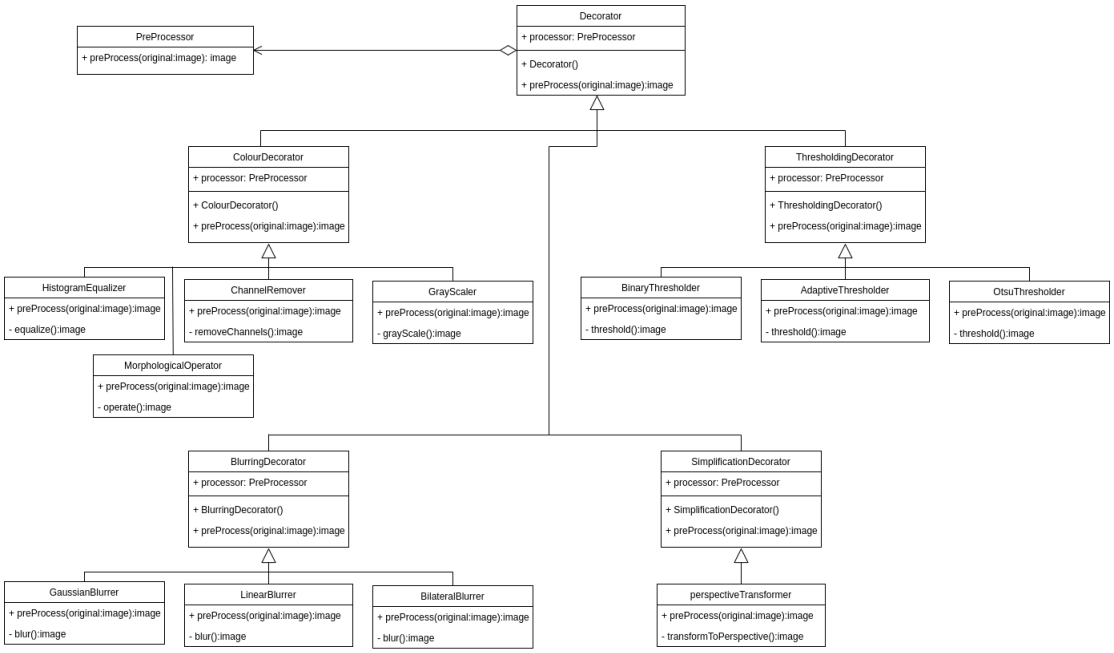


Figure 2: Image Pre-Processing Class Diagram

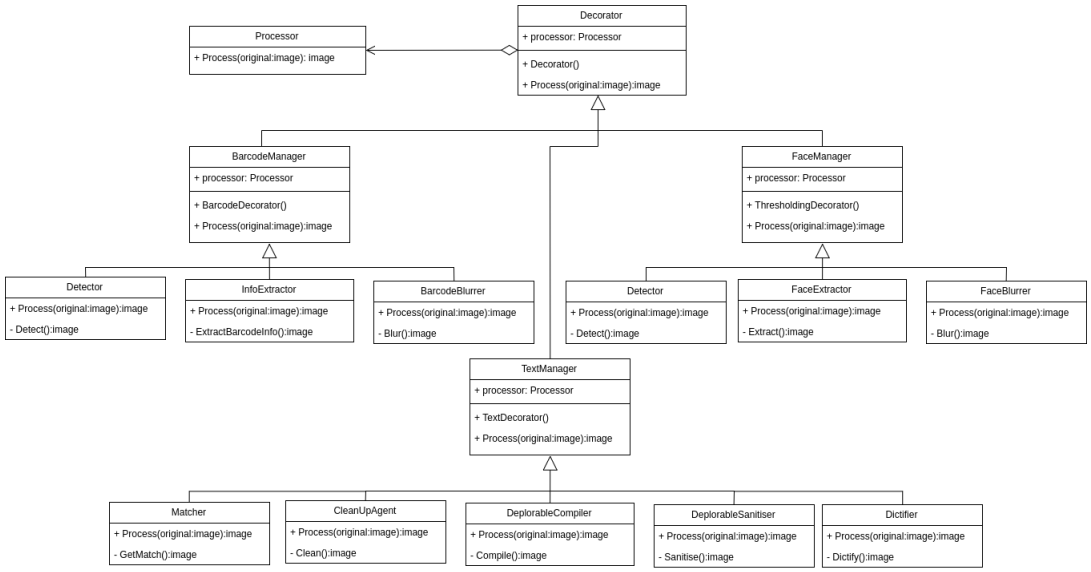


Figure 3: Image Processing Class Diagram

3.1.2 Sequence Diagram

3.1.3 Activity Diagram

3.1.4 State Diagram

3.1.5 Use Case Diagram