**Computer Vision for Assembly Line**

**(CVAL)**

**Software Design Description**

**TABLE OF CONTENTS**

1. Introduction

1.1. Purpose

1.2. Scope of the project

1.3. Definitions, acronyms and abbreviations

1.4. References.

1.5. Audience

2. Definitions

3. Design Description and Information Content

3.1 Introduction

3.2 Design stakeholders

3.3 Design Views

3.3.1 Technical Platform

3.3.2 Image Processing

3.3.3 Data Structure Design

3.3.4 Performance

3.4 Design Viewpoints

3.4.1 Context Viewpoint

3.4.2 Functional Viewpoint

3.4.3 Information Viewpoint

3.4.4 Deployment Viewpoint

3.4.5. Operational Viewpoint

3.4.6. Interface Viewpoint

3.5 Design Elements

3.5.1 Design Entities

3.5.2 Design Relationship

4 Design Viewpoints

4.1 Context Viewpoint

4.2 Functional Viewpoint

4.3 Information Viewpoint

4.4 Deployment Viewpoint

4.5. Operational Viewpoint

4.6. Interface Viewpoint

5.Deployment diagram

6.Use Case Realizations

**1. Introduction**  
**1.1. Purpose**



The aim of the document is to give a written description of the design of our software and overall guidance to the architecture of the software project. The basic purpose of this SDD document is to provide our development team a stable reference, outline all parts of the software and how they work. It defines how our developer team see the product and its functionality.

**1.2. Scope of the Project**

This software system will be providing computer vision to an assembly line in the die manufacturing unit of client’s company. However, the system provided can be considered general and will be implementable in the assembly line for any other manufacturing unit. This system is designed to maximize the productivity of the client’s manufactures by providing a tool to detect defects in the manufactured goods on the assembly line which would otherwise have to be performed manually.

More specifically this system will be designed to set up a connection with a camera placed as a part of the assembly line which will be clicking pictures of the manufactured products. The pictures would then be compared with the pictures of the ideal manufactured product. If the relative error is higher, the manufactured product can be discarded else it is safe to go on. The scope of this software however is limited to scanning only the parts of the product which can be captured by the camera for errors. The parts of the product like the bottom surface kept on the conveyor cannot be scanned. An automated mechanical system can be utilized for discarding the defected product or else the process can be manual too.

**1.3. Definitions, acronyms and abbreviations**

|  |  |
| --- | --- |
| Operator | The person responsible for removing the defected objects |
| Products | The manufactured goods that come out of the assembly line |
| CVAL | Computer Vision for Assembly Line (i.e., our software product) |
| Design View | A representation comprised of one or more design elements to address a set of design concerns from a specified design viewpoint. |
| Design Viewpoint | The specification of the elements and conventions available for constructing and using a design view. |

**1.4. References**

* A concise Introduction to Software Engineering, Pankaj Jalote
* Wikipedia(www.wikipedia.com)
* IEEE ([www.IEEE.org](http://www.IEEE.org))

**1.5. Audience**

The primary clients are industries which use die assembly line production. Such industries need precise products. We can extend this software’s use in the future for all industries using assembly line. Secondary actors consist of all the stakeholders of the system, including a worker of the die industry.

**3. Design description information content**  
**3.1. Introduction**



The deployment diagram describes structures that reside within the software and tells us about the relationship of the data objects.

The architecture design uses information flowing characteristics, and maps them into the program structure. It tells us about the audience, design concerns and views, technical platform, data structure design and performance of software.

The interface design describes internal and external program interfaces, as well as the design of human interface. It describes points like visibility of system status, consistency and standards, flexibility and efficiency of use, aesthetic and minimalist design.

Use case realization describes how a particular use case is realized within the design model, in terms of collaborating objects. It describes points like capturing an image, analysis of product, stage errors.

**3.2 Design stakeholders**

• Developers

• Design Team

• Client

**3.3 Design Views**

It describes the physical model that decomposes the software into components. Each component is described in terms of its external interfaces and the dependencies on other components, this in order to allow the programmers in the next phase of the project to work in parallel. This section describes the software requirements and objectives that have some significant impact on the architecture.

Various modules of the software are loosely coupled but have high cohesion.

The views used to document the CVAL in current phase are:  
**3.3.1 Technical Platform**CVAL will run on C++ using OpenCV.  
OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. OpenCV is written in C++ and its primary interface is in C++.

**3.3.2 Image Processing**

OpenCV has built in libraries which have image processing capability.

**3.3.3 Data Structure Design**

The different angles of the perfect product (die) are captured i.e. all the asides except bottom. These images are stored as a file and are continuously compared with the different snapshots of the images captured. The pictures are stored in a memory buffer which is our hard disk in this case.

**3.3.4 Performance**

Processing a batch of 50 items should take under 1 minute for ideal conditions. Other factors depend on hardware used.

**3.4 Design Viewpoints**

**3.4.1 Context Viewpoint**

Describes the relationships, dependencies, and interactions between the system and its environment

**3.4.2 Functional Viewpoint**

Describes the system’s functional elements, their responsibilities, interfaces, and primary interactions.

**3.4.3 Information Viewpoint**

Describes the way that the architecture stores, manipulates, manages, and distributes information.

**3.4.4 Deployment Viewpoint**

Describes the environment into which the system will be deployed, including capturing the dependencies the system has on its runtime environment.

**3.4.5 Operational Viewpoint**

Describes how the system will be operated, administered, and supported when it is running in its production environment.

**3.4.6 Interface Viewpoint**

Describes how the system will interact with environment and the handling with interaction of the software by the users.

**3.5 Design Elements**

**3.5.1 Design entities**

1) Visual component-Camera and display handling.

2) Processing element

3) Logging utility

**3.5.2 Design Relation**

Visual receives and passes data to processing element. Processing element carries operations on received data and returns to visual component and logging utility.

**4. Design viewpoints**



|  |  |
| --- | --- |
| **4.1** [**Context**](http://www.viewpoints-and-perspectives.info/home/viewpoints/context) | The deployed software is run on a computer system which controls the camera. These cameras are placed in multiple angles near the assembly line and the image quality depends primarily on the camera and the angle at which these photos are taken. |
| **4.2** [**Functional**](http://www.viewpoints-and-perspectives.info/home/viewpoints/functional) | The camera clicks photos of the product from multiple angles. These images are accessed by software and it checks whether the product is defective or not by comparing with a predefined image which is considered ideal by the company. If the product is defective, id of product is flashed on the screen and stored in the log. If the whole batch is found defective, a message to the operator is sent informing him that the whole batch of products is defective. If the product is not found defective, it goes further for processing. |
| **4.3** [**Information**](http://www.viewpoints-and-perspectives.info/home/viewpoints/information) | This software needs to access images and use them to create logs so as to identify any error in the making of the product.  The ideal images of the product which is to be made is provided by the company in the form of a file. The images’ clicked by the camera is stored as a temporary batch file of images with a maximum number of pictures being 50. The logs created after analyses are stored separately in a log file. |
| **4.4** [**Deployment**](http://www.viewpoints-and-perspectives.info/home/viewpoints/deployment) | The cameras’ will continuously capture images of the products on the assembly line Thus software needs basic network access to each camera i.e. each camera can send information to the software for analyses. The computation should be fast enough to analyze each image of the product before the next set of images arrive and store it in the file and update the logs. |
| **4.5** [**Operational**](http://www.viewpoints-and-perspectives.info/home/viewpoints/operational) | Our software is autonomous software which uses the language C++ and implements the OpenCV library. The basic requirements of the system on which the software runs is; a compiler with integrated OpenCV library which can run a C++ code. Being autonomous, once the software starts working no separate operations are needed unless the system crashes. Such case can arise when the graphic driver is not able to link with the images provided. So we would need systems with high graphic processing abilities. |
| **4.6 Interface** | The system comprises the interaction of the images from our image file of the ideal product with the image captured. The defected products from the assembly will be displayed the product ids of the products which are to be removed. The camera capturing the images of the product is provided an interface by CVAL as it can then receive these images and carry on the further processing. There are some criteria that we need to apply to our software in order to have effective, efficient, easily understandable and usable interface.  These are –   1. **Visibility of system status:** The system should always keep workers informed   about what is going on, through appropriate feedback within reasonable time.   1. **Consistency and standards:** Workers should not have to wonder whether the product is falsely shown as defected. 2. **Flexibility and efficiency of use:** Faster systems and assembly line may often speed up the interaction for the software such that the system can cater to both inexperienced and experienced workers. 3. **Aesthetic and minimalist design:** Dialogues should not contain information which is irrelevant or rarely needed. |

# 5. Deployment Diagram



**6. Use Case Realizations**  
Use case realizations-



**6.1.** **Capturing an image.**

• Product comes from assembly line.

• Using camera, images are clicked in different angles.

**6.2**. **Analysing product.**

• Captured image, stored batch wise, intoa file.

• Image analysed by software for defects in product.

• ID of the defective product given in defect log file.

**6.3**. **Stage errors**

• High percentage of products in batch found to be defective

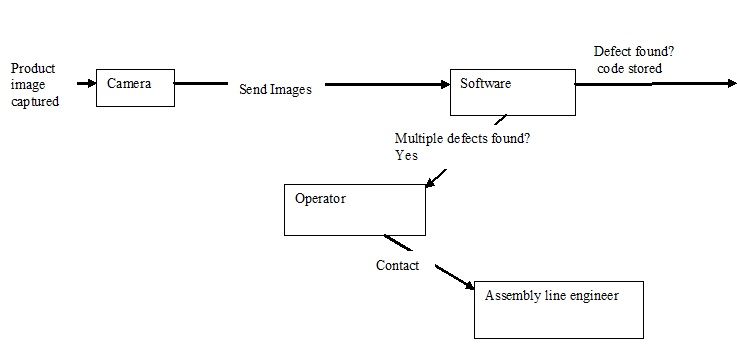
• Contact assembly line engineer in this case.

**4.** **Authorization**

• Operator types in the given id and password

• System created log file having defects listed in order of detection is shown.

**6.4 Diagram**

****