*University Polytechnica of Bucharest*

*Faculty of Engineering in Foreign Languages*

*Software Engineering Master, year II, group 1272E*

Software Design Document

“Voting-Based Image Binarization”

**Team:**

**Barbu Sabina**

**Poșircaru David**

**Radu Ioan Cosmin**

Table of Contents

[1 Document purpose 2](#_Toc371633926)

[2 Objectives 2](#_Toc371633927)

[3 Document overview 2](#_Toc371633928)

[4 Data design 2](#_Toc371633929)

[4.1 Linking 2](#_Toc371633930)

[4.2 Temporary 3](#_Toc371633931)

[4.3 File formats 3](#_Toc371633932)

[4.4 Database description 3](#_Toc371633933)

[5 Architectural design 3](#_Toc371633934)

[5.1 System architecture 3](#_Toc371633935)

[5.2 Architectural patterns 4](#_Toc371633936)

[5.3 Architecture diagram 5](#_Toc371633937)

[5.4 Implementation requirements 5](#_Toc371633938)

[5.5 Component interaction 6](#_Toc371633939)

[6 User interface design 6](#_Toc371633940)

[6.1 Flow chart 6](#_Toc371633941)

[7 Testing issues 7](#_Toc371633942)

[7.1 Critical components 7](#_Toc371633943)

[7.2 Alternatives 7](#_Toc371633944)

# Document purpose

This software design document describes the data, architectural and user interface design of the “Voting-Based Image Binarization” project, containing the software structure and components necessary for implementation.

# Objectives

The main objective of this document is to provide roadmap for the developers, offering technical details about the way the project should be implemented.

# Document overview

Chapter 1 states the document purpose, and Chapter 2 the document's objectives.

The remainder of this document consists of 4 more chapters.

Chapter 4 offers a perspective of the data design of the application. It covers the linking between data, the file formats used.

Chapter 5 covers information about the architectural design for the application, with the architectural patterns used, a defining diagram and the component interaction.

Chapter 6 shows details abut the design of the user interface, the flow chat through the application.

The last chapter, Chapter 7 presents the testing issues that can be encountered for the binarization application.

# Data design

## Linking

Because the project consists in two parts: a “Binarization Algorithm Module” (BAM) and a “Voting Binarization Algorithm Module” (VBAM) we have to consider the way the two modules are linked.

The “Binarization Algorithm Module” (BAM) output is a 1bpp image, output\_image and an 8bpp image, output\_image-confidence. The first image is the actual binarization and the second a gray-scale image containing the confidence for the binarization for every pixel. 0 means that the respective pixel was randomly assigned a color (black or white); 255 means that the algorithm is absolutely certain that the respective color of the pixel is correctly assigned.

This output will be then given as input for the “Voting Binarization Algorithm Module” (VBAM). In its turn, the VBAM module will create as output an image, performing "educated voting " from the input ones.

## Temporary

There are no temporary files because the output of the “Binarization Algorithm Module” (BAM) will be saved permanently for verifications even after it is given as input for the “Voting Binarization Algorithm Module” (VBAM).

## File formats

The following files are used in the program:

* The input file for the “Binarization Algorithm Module” (BAM),which is a Bitmap image (bmp extension).
* The two images which are outputs of the BAM , also in Bitmap format
* the input of the “Voting Binarization Algorithm Module” (VBAM) is also in Bitmap format.
* The output VBAM is in Bitmap format also.

## Database description

The project does not need a database as the images will be saved in folders and will have suggestive names.

# Architectural design

## System architecture

The project has in two parts: a “Binarization Algorithm Module” (BAM) and a “Voting Binarization Algorithm Module” (VBAM).

The “Binarization Algorithm Module” (BAM) is an executable which receives as input from the command line two file names: input\_image and an output\_image. The BAM will return an error code: zero for no error and nonzero in case an error occurred, specifying the error type.

The “Binarization Algorithm Module” (BAM) output is a 1bpp image, output\_image and an 8bpp image, output\_image-confidence. The first image is the actual binarization and the second a gray-scale image containing the confidence for the binarization for every pixel. 0 means that the respective pixel was randomly assigned a color (black or white); 255 means that the algorithm is absolutely certain that the respective color of the pixel is correctly assigned.

This output will be then given as input for the “Voting Binarization Algorithm Module” (VBAM) in case the error code is zero, meaning there were no errors, otherwise the result won’t be considered. The VBAM can receive as input any number of BAM outputs and performs an “educated voting”. This means that VBAM can take into consideration things as the number of images that say a certain pixel is white or black and/ or a weighted decision based on the confidence image.

## Architectural patterns

Batch sequential processing is a sub-type of data flow architectural style. In addition to the common features of the data flow architectural style’s computational model, in the batch sequential style, components are independent programs. They are executed sequentially, i.e. one component runs to completion before the next starts. This is the case for our Binarization System, BAM must run tot ocmpletion before VBAM can start.

The data is transmitted between components as a whole batch rather than a stream of data elements.

Batch sequential processing architectures inherit almost all advantages of the data flow architectural style, especially the reusability and modifiability. In this ways, we avoid complicated issues related to the synchronisation between components by   
executing them one after another.

On the other hand, because the components of the Binarization System do not execute concurrently, the performance of the system may be less satisfactory.

## Architecture diagram

The architecture diagram is depicted in Fig. 1:

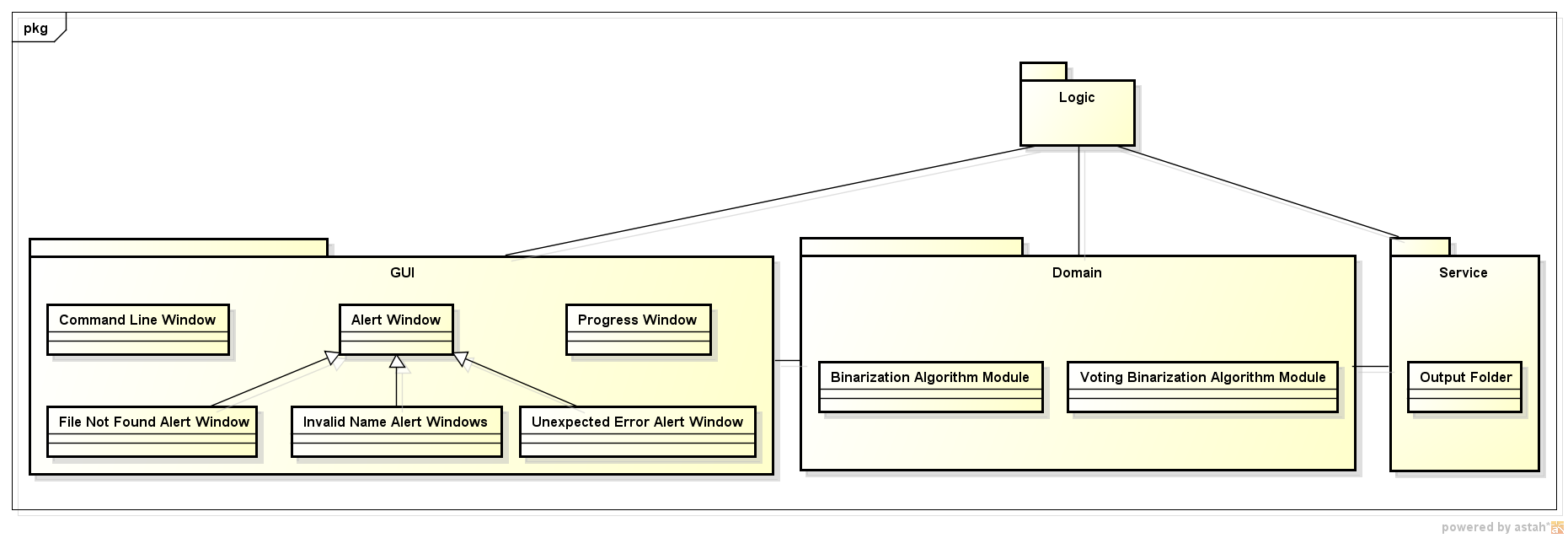


Fig.1 Architecture diagram

The architecture consists in three main parts:

* a graphic user interface (GUI),
* the domain
* the service part with the logic part supervising them.

In the GUI we support the command line input for the *input\_image* and an *output\_image* names which are introduces by the user. Also in the GUI we alert the user in case errors appear, such as file not found, invalid names given by the user and other unexpected errors.

In the domain the algorithms for “Binarization Algorithm Module” (BAM) and for “Voting Binarization Algorithm Module” (VBAM) are implemented.

The service represents just the output folders in which the algorithms in the domain logic should write.

## Implementation requirements

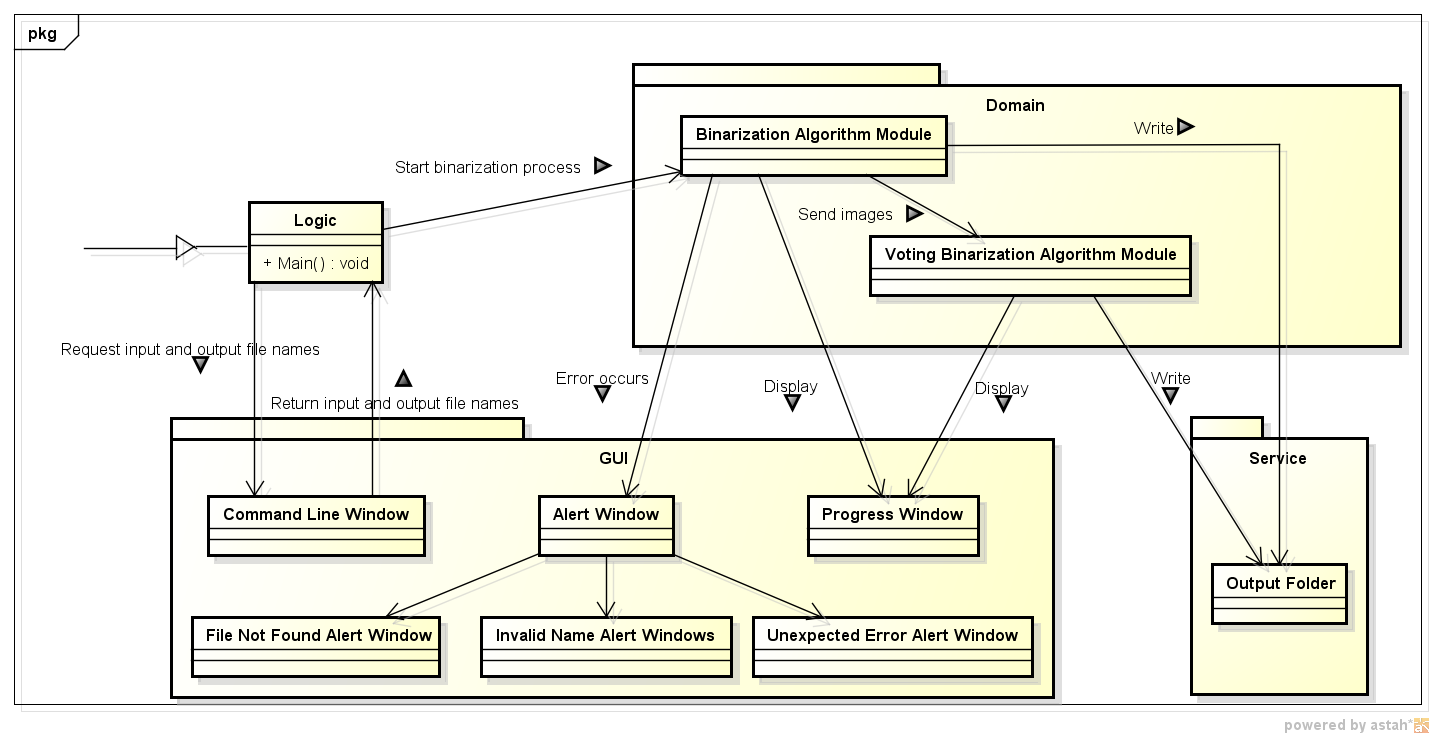
The implementation requires a binarization algorithm to be able to run on the machine.

In terms of resources, hardly any algorithm requires more memory than what is usually available for modern commodity computers.

However, because sometimes execution time can spiral out of control, the processing involved is usually easy to parallelize so more hardware and efficient implementations should normally solve, at least partially, this problem.

## Component interaction

Fig.2 shows the components interaction:

Fig.2 Component interaction diagram

# User interface design

## Flow chart

The user interface flow start from a a command line where the user must input two file names: *input\_image* and an *output\_image*. Fig.3 shows more details.

The flow continues with the Progress Window, that shows the user starting from the binarization call the state of the binarization during the process.

Also alert windows are displayed in case of errors, such as file not found, invalid names given by the user and other unexpected errors.

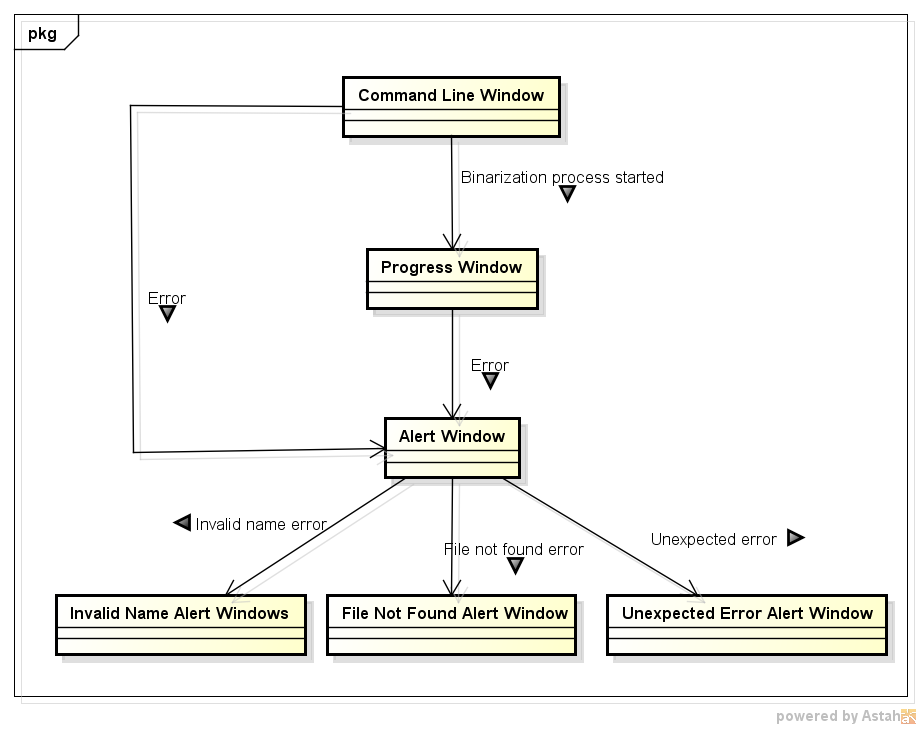


Fig.3 Flow chart diagram

# Testing issues

## Critical components

The most important components of the system are the two main components “Binarization Algorithm Module” (BAM) and “Voting Binarization Algorithm Module” (VBAM).

For this components, testing must be performed before introducing them in the application, as The VBAM component relies heavily on the BAM ones in order to be able to produce the voting algorithm correctly.

Also, integration testing must be performed on the whole system, to see that the two modules interact with each other and obtain the desired result.

## Alternatives

In case something goes wrong in the BAM module, the VBAM will not start, because it will wait for the images to be produces, the binarized images.

If there is an error in the BAM module, the binarization will not be performed, so the VBAM module will not be affected by this.

The alternative path is this, signal an error code for any error in BAM, and the VBAM will only perform voting when the error was solved. The normal flow is to produce the image in the BAM and send it as input to VBAM.