UNIVERSITY OF GOTHENBURG

Automatic classification of UML Class diagrams through image feature extraction and machine learning

Jóel Hjaltason
Ingimar Samúelsson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2014

Automatic classification of UML Class diagrams through image feature extraction and machine learning

Jóel Hjaltason
Ingimar Samúelsson

Examiner: Matthias Tichy

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2014

Abstract

Unified Modeling Language (UML) Class diagrams (CD) are a large part of the software development industry in relation to design. To be able to research UML, academia needs to have access to a database of UML diagrams. For building such a database, automatic classification of UML diagrams would be very beneficial. This research is of a design nature, and focuses on investigating CD classification: what features set them apart from other similar diagrams; how these features can be extracted through image processing; and what kind of accuracy is achievable with said features, using the Support vector machine (SVM) algorithm, and comparing it to several different machine learners (ML). The extracted features that this paper proposes for classification -- in conjunction with the chosen ML -- returns, on average, over ninety percent accuracy in classifying UML Class diagrams.

1 Introduction

In software development, UML CDs are used to design and illustrate the structure of software. It's a very important tool when an engineer needs to understand the basic structure of a system -- e.g. when a new engineer, that is unfamiliar with a system, needs to maintain it. They are becoming ever more prevalent within industry and academia -- where model-driven development is becoming a common practice -- and it can be said that they have become an integral part of it. For the furtherment of research and development of UML CDs, images of relevant UML diagrams from industry and academia need to be gathered into a database. For this to be possible it is necessary to have automatic classification of CD images. In this paper we propose a CD classifier that determines if an image is a CD or not. The classifier operates by extracting relevant information about the image and processing that information with a machine learner.

1.1 Purpose of study

The purpose of this study is to aid academic researchers in the accumulation of relevant UML diagrams. It is important for researchers to have an overview of UML usage, knowing in what direction it is going, in relation to development-tool usage and trends. To achieve this, they need to have access to a vast repository of images that represent design diagrams from the industry and academia. This study will focus on researching methods in image feature extraction, for automatic classification of UML CDs. Additionally, the hope is, that it might be possible to add on to this research, making it possible to detect other types of diagrams -- of interest to researchers in academia -- such as UML Sequence diagrams.

1.2 Research questions

This paper will focus on answering one main research question and four sub-questions: How can classification of UML diagram images be automated?

1. What features in an image can be help classify an UML Class diagram, or exclude similar images?
2. How do Support Vector Machines compare to a subset of different machine learning algorithms in classifying UML Class diagrams?
3. What level of accuracy can be expected with said classification?
4. What subset of the proposed features have the most effect in classifying UML Class diagrams?

1.3 Background

Unified Modeling Language [11] is a general-purpose modeling language. It includes various diagrams for use in visualized design within the software engineering field. These diagrams include, in addition to CDs: Sequence Diagrams; Use Cases; and State Machines.

Support Vector Machines [8] are a method of supervised ML. It creates one or more hyperplanes in multidimensional space which is then used for tasks including, but not limited to, classification and regression. The hyperplane that is at the furthest distance from any training data point, provides a functional margin by which to classify a new entry. The wider the margin, the more accurate the classifier.

Hough transformation (HT) [5] is a method for detecting lines and curves in images. Initially, the image is thresholded -- resulting in a black and white image. Then, an external method is used for extracting edges within the image, most commonly the *Canny edge detector* [6] (as is the case here). The outcome of the image thresholding and edge detection is then the foundation that HT builds upon. The algorithm finds lines by transforming the edge image, resulting in two factors that determine the line search. Instead of using *x* and *y*, the position is expressed in the form of the direction *theta* and the distance *r* from the focal point. The focal point that is used, is generally the center point of the image. Edges that are mapped to the same *theta* and *r*, are then considered to illustrate a line.

Suzuki85 (S85) [7] is border-following algorithm for use in the processing of digitized binary images, that uses a sort of topological analysis. It is an extension to a previous border-following algorithm [9] that discriminates between the outer borders and hole borders of a binary image. The extensions are: putting a unique mark on each border, rather than adopting the procedure for every border; and adding a procedure for obtaining the parent border of the currently followed border. The image is scanned and when a pixel (x,y) is found that satisfies the condition for the border-following starting-point, it is assigned an identifiable number. The found border is then followed from the starting point and the pixels on the border are marked.

Ramer–Douglas–Peucker (RDP) [10] is an algorithm that produces a polygon with a small number of edges for arbitrary two-dimensional digitized curves. A curve segment is approximated by a straight-line segment that connects its start and end points. The purpose of this is to find a comparable curve with fewer points -- resulting in a subgroup of points that determine the original curve.

OpenCV [11] is an image processing library that has C, C++, Python and Java support and runs on multiple operating systems. It has many purposes and can be used in various image processing tasks, including implementations of image processing algorithms.

Magick++ [13] is a C++ library that is part of the ImageMagick image-processing library. It is a multipurpose library with various features related to image processing.

Weka [14] is a machine learning software developed at the University of Waikato, New Zealand. It includes tools for testing and visualizing various machine learning algorithms.

1.4 Criteria for success

This research aims to fulfil three success percentages: first, the tests should result in 90% average overall accuracy on classifying an image; secondly, the success rate on discriminating negative images should be, on average, 95%; and thirdly, CDs should be identified with 85% accuracy, on average.

1.4 Outline of thesis

This paper is organized as follows. In section 2 the research methodology is described and discussed and in section 3, related work is covered. Then, an overview of the classifier is described in section 4, and section 5 dictates the approach for implementing the classifier, with the design of the software illustrated in section 6. In section 7, the classifier is validated through image collection and testing, and the results of the research are then put forth in section 8. Finally, section 9 and 10 include discussion and conclusion, respectively.

2 Methodology

We will employ a design research methodology [21] and build upon existing techniques of image processing and feature extraction, to develop a solution to a problem that is previously unaddressed, as we can see in section 3 - related work. We will build a repository of preliminary test-cases and experiment with techniques to process them, extracting the desired information. Lastly, we will evaluate the prototype, producing quantitative benchmarks and statistical analysis.

The requirements of this research can be described with one underlying theme; a computational program that takes in a path to an image and classifies that image as a CD, by returning true or false. There are two non-functional requirements that have to be addressed in this research: limitations on computation time; and success ratio in classification.

To address this research problem, other previous work will be investigated to gain insight into the status of this problem within the field. Experiments will then be done on various types of features -- in an iterative fashion -- that can be used to classify CDs, in an effort to find the most useful features.

3 Related work

In recent years, research has been conducted on detecting diagram features, varying in method and approach. This section will focus on investigating prior research on the subject, mainly focusing on the objectives of said research and comparing them with the objective of the research put forth in this paper.

In [1], the authors propose a tool for processing finalized computer images and extracting CD features from the image. The objective of this is to take in images (such as JPEG) that represent CDs and translate them into XMI format, for use in UML CASE tools. To that end, the feature extraction in this research focuses on specific elements of the diagram -- in addition to the common characteristic between multiple CDs diagrams -- like identifying the role of each element in the model, and types of relations between the elements. This differs from the research that this paper covers because, when classifying if an image is a UML CD or not, eliminating features also need to be detected, and less focus needs to be put on identifying specific roles or relations between elements. The type of relation is not as important, but the relation itself is important.

In [2], the authors propose a method for online recognition of hand drawn UML diagrams. The system takes input from electronic devices and diagnoses pen strokes (hand movements) into UML elements -- the features being detected based on the movements. The research differs from what this paper covers, since [2] focuses on detecting the elements as they are being drawn, as opposed to detecting them in finalized images.

In [3], the authors propose a system for recognizing and automatic learning of sketched graphic symbols in engineering drawings. The objective of this research is to combine pattern recognition techniques with machine learning concepts in order to be able to learn and recognize new symbols in engineering diagrams. This research differs from the one in this paper, since the system takes images representing diagrams and finds symbols in them, as opposed to taking in any type of image and classifying if set image is of a specific diagram type.

In [4], the authors present a method for converting network-like image-based engineering diagrams into engineering models. It takes in either sketches or computer generated images that represent engineering diagrams and converts them. They focus on being able to take in various types of diagrams, including UML. The image processing and feature extraction has similarities -- to the one proposed in this paper -- in finding diagram elements and relations between them, but, as is the case with [1], the difference lies in the fact that they are not looking for discriminating factors in the image -- the assumption is that the image represents a diagram, and the objective is to transform those features, instead of classifying them.

4 Overview of the classifier

Diagrams come in all shapes and forms (Figure 1), and it is important to distinguish between them -- to be able to classify the right diagram images as UML CDs. For this reason, it was important to look at not only CDs, but also other different but similar diagrams such as Entity–relationship models (E/R), UML Sequence diagrams and Flowcharts amongst others, when finding the right features.

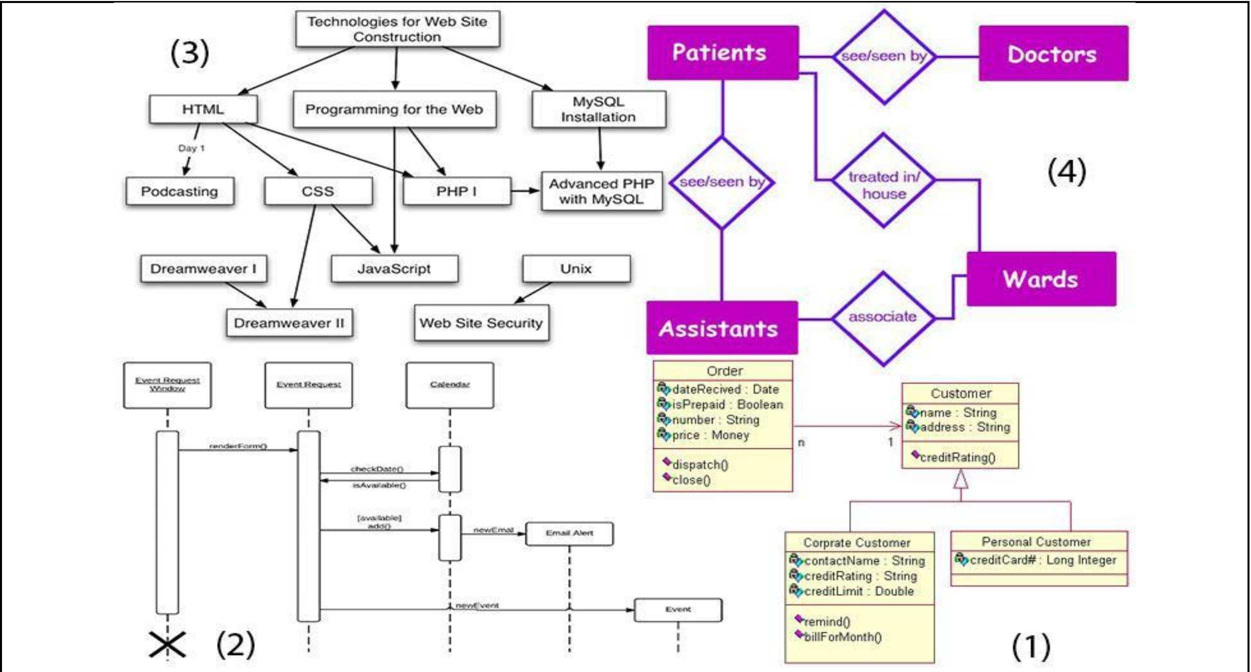


Figure 1: (1) UML Class diagram, (2) UML Sequence diagram, (3) Flow chart, (4) E/R model

CDs have three key factors that can be used to describe them: (1) they consist of classes, in the form of rectangles; (2) the classes are related to each other in the form of connecting lines; and (3) the classes are divided into sections with the name of the class, attributes and operations. The 3rd describing factor is, though, not general. It does not apply to all classes within the diagram, but in almost all UML CD there are classes divided in this manner. As can been seen in Figure 1, the 1st and 2nd of the defined characteristics of UML Class diagrams can apply to many types of diagrams or charts. Because of that it was also important to extract more information from the image, than only information that is descriptive of CD. As a result, other geometrical shapes had to be extracted as well: ellipses; rhombuses; and triangles. That information would then also be used in the calculation of the image features.

The extracted shapes and lines are used to generate statistical information about the image, which is then used in conjunction with SVM to determine if the image is a CD. The classifier takes in a path parameter to an image, either a URL or a folder path and returns true or false. If the input file is not found or corrupted, exception is not outputted, but instead, it returns false. It was implemented in this fashion, because the program would be used with a web crawler and exception handling was not considered necessary.

5 Approach

To be able to classify an image, two things had to be done. First, shapes and lines from the image had to be extracted and secondly, the gathered information had to be transformed into features that could be used to determine if an image is a CD or not. Additionally, to avoid prolonged processing time on complex photographs, images have to pass a pre-check before being processed: the most frequent color in the image has to represent at least 10% of the image; and the image's color-histogram median value must be above 100. Both of these figures were obtained by observation of trends within our dataset. A view of the overall framework can be seen in Figure 2.

OpenCV is used for processing the image, and execution of external algorithms for extracting preliminary lines and shapes. OpenCV does not support all of the desired image formats, so the Magick++ library is used to process the image formats that OpenCV does not support, converting them into JPEG if the format is other than PNG or JPEG. As a result, all the most common image formats [13] are supported.

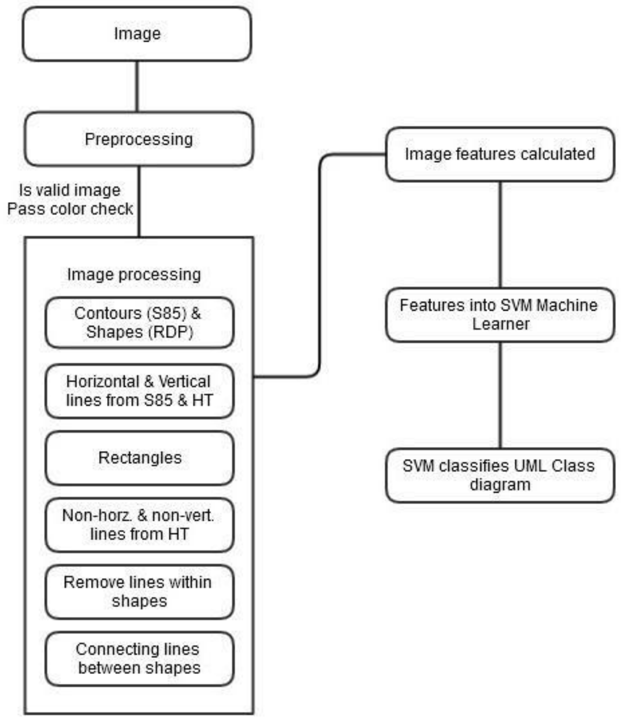


Image
Preprocessing
Is valid image
Pass color check
Image processing
Contours (S85) & Shapes (RDP)
Horizontal & Vertical lines from S85 & HT
Rectangles
Non-horz. & non-vert. lines from HT
Remove lines within shapes
Connecting lines between shapes
Image features calculated
Features into SVM Machine Learner
SVM classifies UML Class diagram

Figure 2: Overall framework

Figure 2 illustrates the different stages of the classification process: preprocess; extraction of shapes and lines; feature calculation; and classification through SVM. This is covered in more detail in the following sections.

5.1 Constraints

Execution time of the image processing was a non-functional requirement. It was decided that the time it took to classify an image should not exceed ten seconds. Some images are more complex than others, so this is defined as a weak constraint -- meaning the ten seconds were to be considered on average in the test set, but not for any given image.

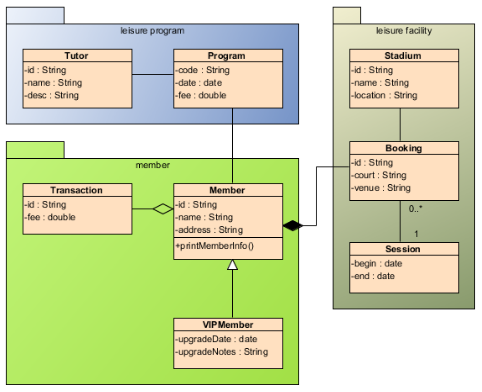
Content comprehension could not be taken into account in this research because of time constraints. That means that textual analysis of the image was not processed and included in the extracted features. The result is that images that have exactly the same features as CDs but differ in textual content, are therefore classified as CDs.

Because this classifier will be used with a web crawler to gather UML CD images into a database, eliminating non-CD images has greater value than including CD images.

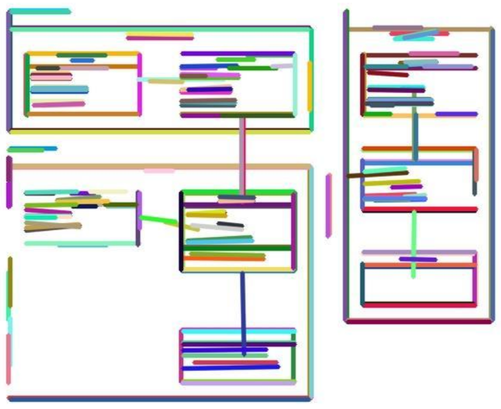
5.2 Image processing

Shape and line extraction is carried out in conjunction with three external algorithms: *Hough transform*; *Suzuki85*; and *Ramer–Douglas–Peucker*. The contours that S85 finds are used to find various shapes and are subsequently broken down into straight lines. By doing this in conjunction with HT it was possible to better assure that as many lines where detected as possible. The lines are then processed, so that horizontal and vertical lines, that are on the same axes and represent the same line, are joined together into a single line. Rectangles that are not caught using S85 are then extracted by finding horizontal lines that are parallel and in the same position on the x-axis, and have the same two vertical lines intersecting them on each end.

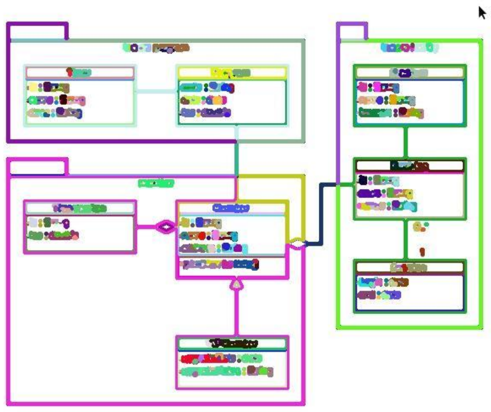
RDP is used to find different types of shapes. If it finds a polygon that might possibly be one of the sought after shapes, a validation takes place. The coordinates are investigated to see if it fits the criteria of any of the shapes: rectangles should have four 89° < x < 91° corners; rhombuses should have four 60° < x < 120° corners; triangles should have three 40° < x < 100° corners; and in case of ellipses, all of the points in polygon need to fit into the ellipse equation (Appendix B).



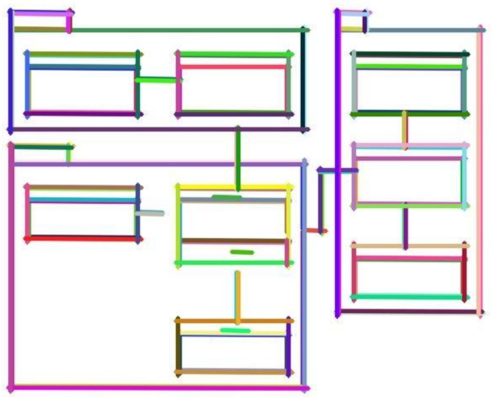
A. Original image



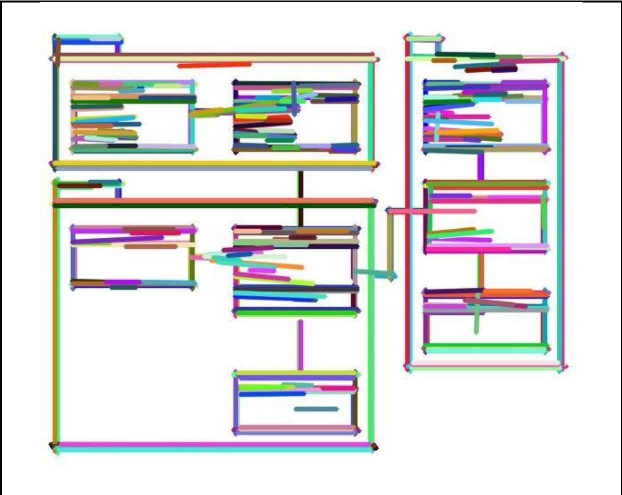
B. Hough transform



C. Contours



D. Joined lines #1


E. Joined lines #2

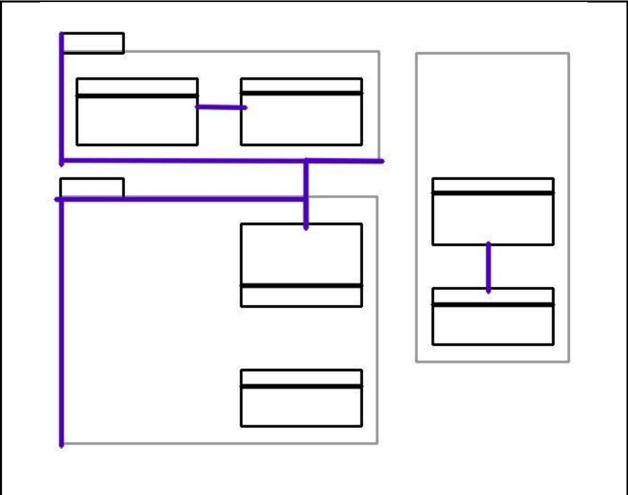

F. Extracted elements

Figure 3: Image process

Figure 3 shows the basic steps of the image process. This picture was chosen because it is a good example for demonstrating why lines were joined between the two algorithms. As can be seen in picture B in Figure 3, with HT, many of the rectangle lines are not extracted, or the extracted lines are segmented and/or incomplete. Such lines make it very difficult to find the rectangles in the image. S85 returns an unlimited amount of points in each contour -- multiple lines segments, in other words. The extracted contours from S85 can be seen in picture C. By examining that picture, it is apparent that the algorithm catches more of the lines than HT -- it does not miss any of the desired lines, because it seems to catch everything in the image. The lines are joined in three phases: (1) the contours that are found are split into lines, and horizontal and vertical lines are extracted; (2) horizontal and vertical lines that HT finds are collected and joined with (1); and (3) lines, found by HT, that are not vertical or horizontal are collected and joined with (1). After phase 1 and 2 (picture D in Figure 3), rectangles are collected through the method previously explained in this section. After the rectangles have been collected, phase 3 (joining lines; picture E) is conducted, and then all lines within shapes are removed.

It was to be expected, that the algorithm would not be able to catch all rectangles in the image. By comparing the extracted rectangles in the image (picture F in Figure 3) to the original image (picture A), it is apparent that the extraction misses two of the classes in the image. The reason for this is the HT line detection misses important lines. If the rectangles in question are examined in picture B, it can be seen that the sides of the rectangle are missing and, additionally, the detected lines are segmented and incomplete. It could be expected, that these rectangles would be found, based on visually examining picture D -- as is the case with the other rectangles with missing line segments the picture. Based on that image (D), it is hard to explain why those rectangles are missed, not the others, but most likely the joining of the lines did not result in four complete lines that represent a rectangle in x and y coordinates, and within the previously detailed rectangle restriction.

To find connecting lines between various shapes, lines that intersect said shapes are gathered. These intersecting lines are then followed until another intersection is found or to the end of said line. If another intersection is found, the line is added as a connecting line between the two intersected shapes -- if a connecting line between those two shapes does not exist. As can be seen in picture F in Figure 3, the connecting lines do not follow the shortest distance between the rectangles. When the search for connecting lines is conducted, all directions are followed until the line's end is reached, or a connection is found. The first route that is found between the shapes is the one that is added. This was done intentionally, because finding the shortest distance was not considered important in the image processing, since the extracted features only involved the percentage of connections -- the distance was not calculated.

Color distribution is gathered by examining all the pixels in the image. Colors are grouped together based on their RGB colors, and to join similar colors, the 255 RGB range is divided by 15, resulting in 17 to the power of 3, color combinations. The reason for joining similar colors was to get the background color of image as one color, and the number 15 was used because the numbers were integers and dividing 255 by the chosen number had to result in an even number.

A native function of OpenCV is used to extract the histogram of all images, and the median value is found by the use of a simple loop system that first calculates the sum of the values, and then loops again to find the position of that sum divided by two.

5.3 Feature extraction

It was considered important that the extracted features would not be affected by unrelated information. Information that does not discriminate CDs from other diagrams, should not affect the outcome. For example, one CD may have three rectangles and another CD may have 25 rectangles, but in both images the rectangles are likely to cover a similar portion of the image. So instead of counting different extracted elements, the extracted features are represented in the form of ratios and percentages relative to the image. The extracted features, and explanations of the ratios and percentages, can be seen in Table 1.

Feat. no.	**Name**	**Description**
F01	*Rectangles' portion of image, percentage*	Calculated by dividing the sum of the area of all the rectangles with the area of the image itself
F02	*Rectangle size variation, ratio*	Calculated by dividing the rectangle size standard deviation with the rectangle average size
F03-06	*Rectangle distribution, percentage*	The image is divided into four equally sized sections and the area of the rectangles inside the

		sections is then divided by the total area of the rectangles. The 4 sections sum up to 100%
F07	*Rectangle connections, percentage*	Calculated by counting all rectangles that are connected to at least one rectangle, and dividing that number with the total amount of rectangles in the image
F08-10	*Rectangle dividing lines, percentage*	The rectangles are split into three groups, with rectangles that have: no dividing lines (F08); one or two dividing lines (F09); or three or more dividing lines (F10). This produces three numbers that represent the percentage of rectangles within each group
F11/F12	*Rectangles horizontally/vertically aligned, ratio*	Sides of rectangles, horizontal (F11) and vertical (F12), that are aligned with sides of other rectangles are counted. The numbers are then divided with the number of detected rectangles in the image -- resulting in two ratios on rectangle horizontal and vertical alignments
F13/F14	*Average horizontal/vertical line size, ratio*	Average size of horizontal (F13) and vertical (F14) lines that are larger than ⅔ of the images width or height, divided by the images width or height, respectively
F15	*Parent rectangles in parent rectangles, percentage*	Rectangles that have rectangles within them can possibly be packages. This feature is the percentage of the area of those parent rectangles that is within other parent rectangles
F16	*Rectangles in rectangles, percentage*	This feature is calculated in the same manner as F15, but with rectangles, instead of parent rectangles
F17	*Rectangles height-width ratio*	The average ratio between the height of the rectangles and the width of the rectangles
F18	*Geometrical shapes' portion of image, percentage*	The same as F01, but with rhombuses, triangles and ellipses
F19	*Lines connecting geometrical shapes, ratio*	The number of connecting lines from shapes, other than rectangles, divided by the number of detected shapes in the image
F20	*Noise, percentage*	Detected lines that are outside of rectangles, divided by the number of all detected lines

F21-23	*Color frequency, percentage*	Three most frequent colors in the image are found. Then a percentage out of all appearing colors is found for the three colors

Table 1: Extracted features

5.4 Classification

SVM was chosen as a machine learner, for several reasons. The classifier has constraints on processing time, and SVMs are efficient in that respect. They also don’t need large data sets to be trained properly and they can handle complex, non-linear classification.

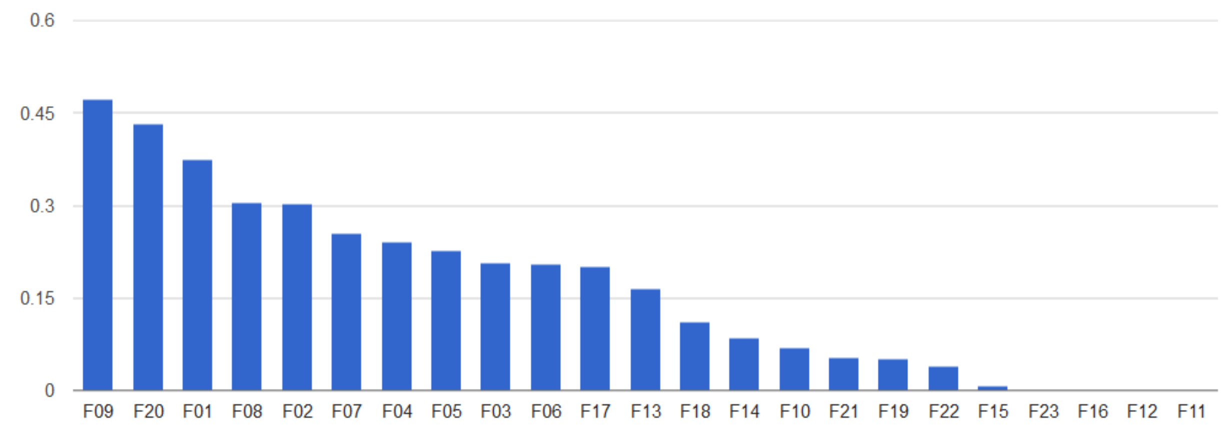


Chart 1: Information gain

Information Gain is the expected reduction in entropy caused by partitioning the examples according to a given attribute. In other words, it tells us how much information can be gained from an attribute, to aid in the accuracy of the ML's classification.

Based on Chart 1, there are four features that have little effect on the classification. As a result, these features were removed and tests were implemented again. But that did not increase the accuracy of classifier, but instead reduced the accuracy in the tests, especially with regards to non-CD images. That resulted in those features being reinstated, because -- as they were intended for -- they helped eliminate false-positives. Those features were created to help eliminate images that had similar features to CDs, but where the rectangles were, for example, aligned or within other rectangles, which should not occur in a CDs.

The most crucial feature is the percentage of rectangles that have one, or two, dividing lines (the 3rd defined feature of CDs). Additionally, the 4th most important feature, for discrimination, is the percentage of rectangles with no dividing line. The second most important feature is noise, also for eliminating purposes. Noise is information that is not within rectangles, and the higher that ratio is, the more it reduces the likelihood of the image being a CD. As expected, since CDs generally have a high rectangle coverage, F01 is one of the three most important features, ranked no. 3. The last of the five most influential features is F02 (rectangle size variation), because images with a high variation in the size of rectangles is less likely to be a CD.

6 Software design

The software is divided into two main sections: machine learning (ML); and image processing (IP). A UML Class diagram of the system can be seen in Figure 4. Within the ML part of the software, there are two actions: train (Class: MLTrainer); and classify (Class: UMLClassifier). As the names suggests, train trains the ML by taking in positive and negative images, and constructs a support vector machine. The classifier takes in an image, or set of images, and classifies them as CDs or not, using the generated SVM. These actions are performed using the OpenCV machine learning library. The ML interacts with the image processing section of the software (Class: ImageProcess). It sends an image's path to the IP and is returned a list of features that represent the image, and uses that for classification. The IP class is the center of the system, interacting with all of the other classes. The class CalcFuns is a static class that performs all of the necessary calculations that happen throughout the image process and feature extraction and is used by most of the other object classes. The other three classes underneath CalcFuns, in the software CD, are object classes that perform the actions that their names indicate. To illustrate their role within the design, OpenCV and ImageMagick libraries are represented as classes within the diagram.

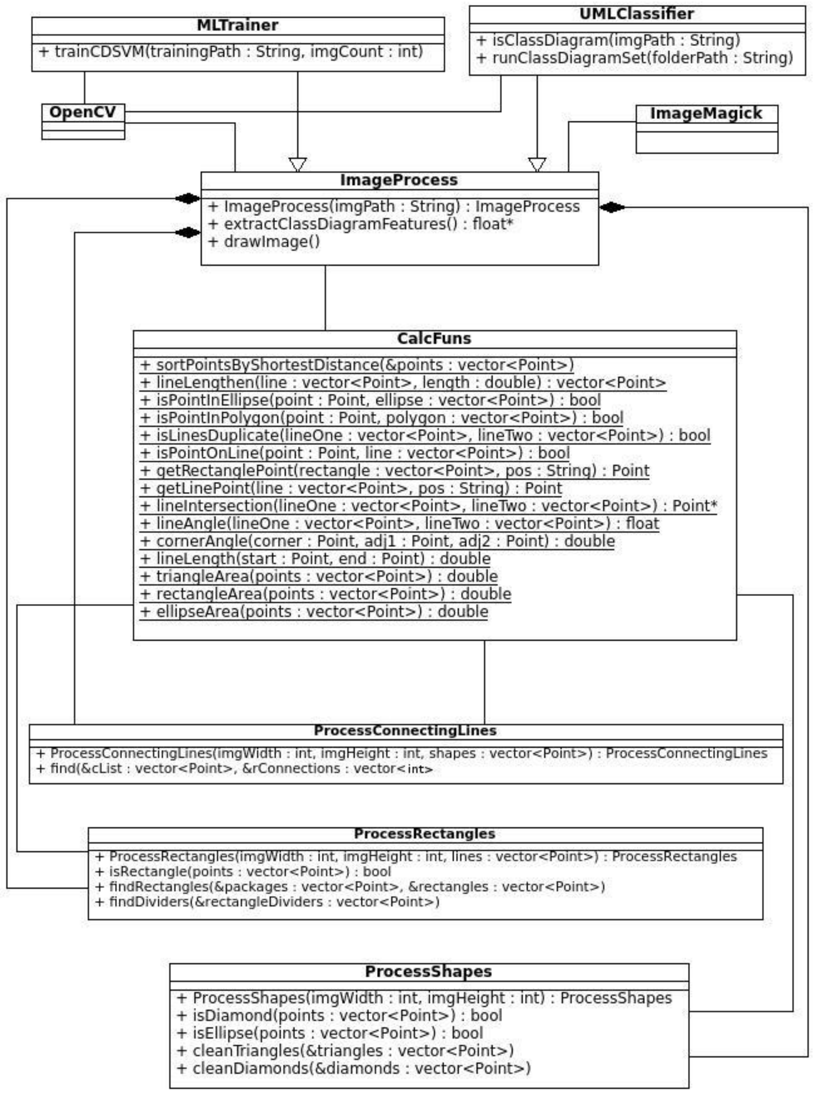

MLTrainer
+ trainCDSVM(trainingPath : String, imgCount : int)
UMLClassifier
+ isClassDiagram(imgPath : String)
+ runClassDiagramSet(folderPath : String)
OpenCV
ImageMagick
ImageProcess
+ ImageProcess(imgPath : String) : ImageProcess
+ extractClassDiagramFeatures() : float*
+ drawImage()
CalcFuns
+ sortPointsByShortestDistance(&points : vector<Point>)
+ lineLengthen(line : vector<Point>, length : double) : vector<Point>
+ isPointInEllipse(point : Point, ellipse : vector<Point>) : bool
+ isPointInPolygon(point : Point, polygon : vector<Point>) : bool
+ isLinesDuplicate(lineOne : vector<Point>, lineTwo : vector<Point>) : bool
+ isPointOnLine(point : Point, line : vector<Point>) : bool
+ getRectanglePoint(rectangle : vector<Point>, pos : String) : Point
+ getLinePoint(line : vector<Point>, pos : String) : Point
+ lineIntersection(lineOne : vector<Point>, lineTwo : vector<Point>) : Point*
+ lineAngle(lineOne : vector<Point>, lineTwo : vector<Point>) : float
+ cornerAngle(corner : Point, adj1 : Point, adj2 : Point) : double
+ lineLength(start : Point, end : Point) : double
+ triangleArea(points : vector<Point>) : double
+ rectangleArea(points : vector<Point>) : double
+ ellipseArea(points : vector<Point>) : double
ProcessConnectingLines
+ ProcessConnectingLines(imgWidth : int, imgHeight : int, shapes : vector<Point>) : ProcessConnectingLines
+ find(&cList : vector<Point>, &rConnections : vector<int>
ProcessRectangles
+ ProcessRectangles(imgWidth : int, imgHeight : int, lines : vector<Point>) : ProcessRectangles
+ isRectangle(points : vector<Point>) : bool
+ findRectangles(&packages : vector<Point>, &rectangles : vector<Point>)
+ findDividers(&rectangleDividers : vector<Point>)
ProcessShapes
+ ProcessShapes(imgWidth : int, imgHeight : int) : ProcessShapes
+ isDiamond(points : vector<Point>) : bool
+ isEllipse(points : vector<Point>) : bool
+ cleanTriangles(&triangles : vector<Point>)
+ cleanDiamonds(&diamonds : vector<Point>)

Figure 4: UML Class diagram

7 Validation

The images that were used in the validation process were collected by using Google image search. The image collection consisted of two separate accumulation phases: collecting images that represented CDs; and collecting non-CD images and images that represented similar diagrams. To search for CDs the phrase 'UML Class diagram' was used. Additionally there were various search strings concatenated with the phrase following them, such as: hospital; transport; and communication. That was done to have a wide variety of CDs and also to be able to accumulate a larger data set, while avoiding duplicates. Images that have features that can be found in CDs and are of a similar nature were considered important, to test the accuracy of the classifier. As a result, various types of diagrams, charts, blueprints and maps were included in the negative image set -- comprising a large portion of the negative images. To find these images, various search strings were constructed and used in the image search -- including, but not limited to: diagram; blueprint; sequence diagram; chart; flow chart; E/R model; and architectural diagram. It was verified that no duplicates were found in the set -- done by examining the extracted features, visually comparing images with similar features, and removing those that proved to be duplicates. The end-result was a collection of 650 CDs and 650 non-CDs -- 1300 images in total. Samples of the images can be seen in Appendix A, table C and D.

Three different tests were performed on the accuracy of the classifier. The average accuracy of those tests was 92%.

Test	**Overall**	**Negatives**	**Positives**
Test A	93.85%	95%	92.59%
Test B	90.19%	86.15%	94.23%
Test C	92.12%	91.92%	92.31%
Average	**92.05%**	**91.02%**	**93.04%**

Table 2: Classification test results

To test the classifier the image set was split randomly, 60% for training and 40% for testing. Test B is nearly opposite to Test A, where the training and testing set in Test A were swapped and then images were randomly moved from the testing into the training set, to split it 60/40. Finally, Test C was implemented in the same manner as Test A, by randomizing the images again. The results can be seen in Table 2. Further tests are presented later in this chapter.

It is apparent that Test B, relative to the other two tests shows markedly reduced accuracy in classifying non-CDs. Opposingly, Test C had the lowest accuracy in classifying CDs, although Test A wasn't much more accurate. Test B is nearly the opposite of Test A (which shows the best results), which can indicate that the Test A training set has a better set of images to learn from, then Test B has, or that the negative test set in Test B was more challenging for the ML. The average and standard deviation of the positive training set features can be seen in Table A in Appendix A.

8.1 False-positives in Classification tests

This section will examine a sample of 5 false-positive images in the classification tests, focusing on investigating why they were classified as CDs, based on the extracted features and visual examination. The extracted features from those images can be seen in Table B, in Appendix A.

Image #1 (Table B in Appendix A; Figure 5) depicts a diagram structured in much the same way as a CD, and includes two of the three main characteristics of a CD; a large portion of the image is covered by rectangles and the rectangles are connected to each other. The structure is so similar in fact, that if we compare the features extracted from image #1 to the average features of positive training set (Table A in Appendix A), it is within the bounds of the standard deviation in 20 of 23 features. The average deviation of these features from the average of positive training set is approx. 35%. The machine learner can be expected to accept any image that so closely resembles a class diagram. To combat this issue and further develop the classifier, textual analysis of the images would have to be employed.

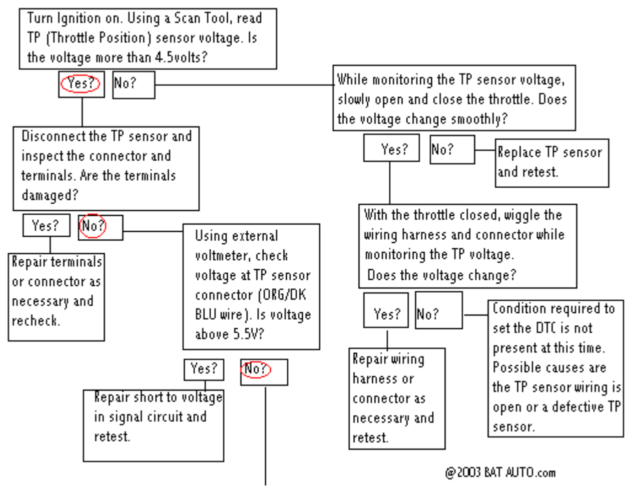

Turn Ignition on. Using a Scan Tool, read TP (Throttle Position) sensor voltage. Is the voltage more than 4.5volts?
Yes?
No?
While monitoring the TP sensor voltage, slowly open and close the throttle. Does the voltage change smoothly?
Disconnect the TP sensor and inspect the connector and terminals. Are the terminals damaged?
Yes?
No?
Replace TP sensor and retest.
Yes?
No?
With the throttle closed, wiggle the wiring harness and connector while monitoring the TP voltage. Does the voltage change?
Repair terminals or connector as necessary and recheck.
Using external voltmeter, check voltage at TP sensor connector (ORG/DK BLU wire). Is voltage above 5.5V?
Yes?
No?
Condition required to set the DTC is not present at this time. Possible causes are the TP sensor wiring is open or a defective TP sensor.
Yes?
No?
Repair short to voltage in signal circuit and retest.
Repair wiring harness or connector as necessary and retest.
@2003 BAT AUTO.com

Figure 5: Image #1

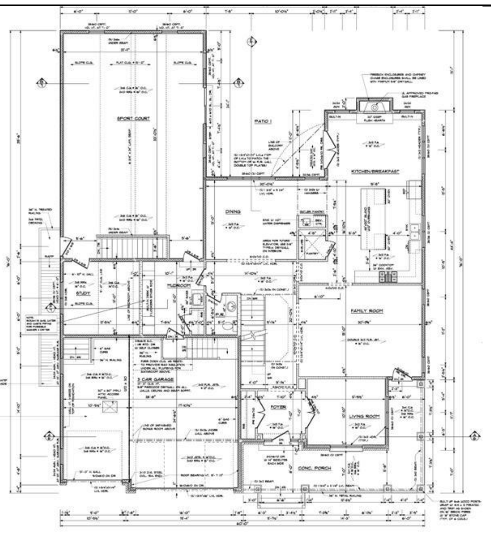


Figure 6: Image #2

Image #2 (Table B in Appendix A; Figure 6) is a blueprint. There is a subset of similar images in the test-set, depicting maps and various types of blueprints. Visually, this image does not resemble a CD, and of the extracted features, 12 where outside of the standard deviation in the positive training set. Additionally, only three of the top ten features in the information gain chart where within the standard deviation. Based on this, it is hard to explain why this image is classified as a CD without an in-depth investigation of the SVM.

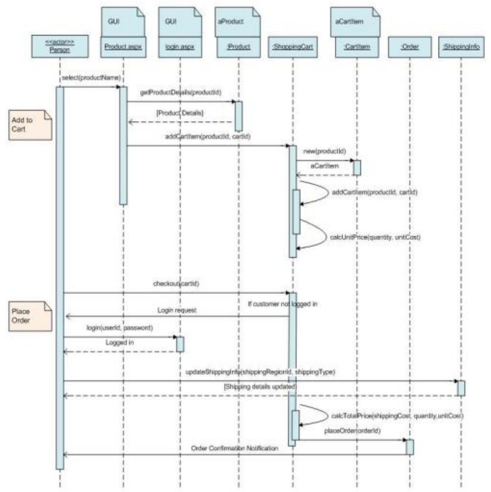

GUI
GUI
aProduct
aCartItem
<<actor>> Person
Product.aspx
login.aspx
:Product
:ShoppingCart
:CartItem
:Order
:ShippingInfo
select(productName)
getProductDetails(productId)
[Product Details]
Add to Cart
addCartItem(productId, cartId)
new(productId)
aCartItem
addCartItem(productId, cartId)
calcUnitPrice(quantity, unitCost)
checkout(cartId)
If customer not logged in
Place Order
Login request
login(userId, password)
Logged in
updateShippingInfo(shippingRegionId, shippingType)
[Shipping details updated]
calcTotalPrice(shippingCost, quantity,unitCost)
placeOrder(orderId)
Order Confirmation Notification

Figure 7: Image #3

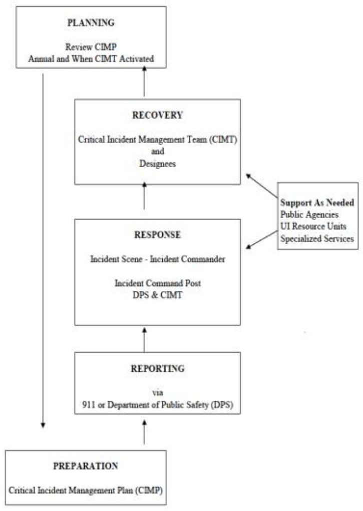

PLANNING
Review CIMP
Annual and When CIMT Activated
RECOVERY
Critical Incident Management Team (CIMT) and Designees
Support As Needed
Public Agencies
UI Resource Units
Specialized Services
RESPONSE
Incident Scene - Incident Commander
Incident Command Post
DPS & CIMT
REPORTING
via
911 or Department of Public Safety (DPS)
PREPARATION
Critical Incident Management Plan (CIMP)

Figure 8: Image #5

Image #3 (Table B in Appendix A; Figure 7) is a sequence diagram. When we examine the extracted features from the image we can see that the most important feature (F09 in Chart 1) is close to being inside the st. dev. of the positive training set. The noise in the image is closely outside of the st. dev. of positive training set. When we look at rectangle distribution (F03-F06; ranked 7-10 in Chart 1) we see that the distribution between the sections are uneven, but still they are all close to being within the st. dev. of training set. The features F01 and F02 are ranked 3 and 5, respectively, in Chart 1 and the image is only 0.3% from the average of F01 (rectangle coverage), which might be one of the reasons why it is classified as a CD. But, when you examine rectangle size variation (F02), it is far outside the st. dev. of the positive training set -- almost double the amount. The result is, that the image does not, visually, resemble a CD, but the extracted features on the other hand do, at times. Of the 650 collected non-CD, 72 were Sequence diagrams.

Image #5 (Table B in Appendix A; Figure 8) resembles a CD. It has six rectangles that cover a large portion of the image and all but one of the rectangles are connected to another rectangle. When we examine the features that were extracted from the image, we see that 5 features (F6; F8; F9; F13; F21) fall outside of the standard deviation of the positive training set. Of these 5 features, 3 are in the top half of the Information gain chart (Chart 1), with F09 and F08 being no. 1 and no. 4, respectively. Based on that it could be expected that the ML would reject this image, but as mentioned before, this image has resembles CD's and a false-positive would always be a possibility without textual analysis.

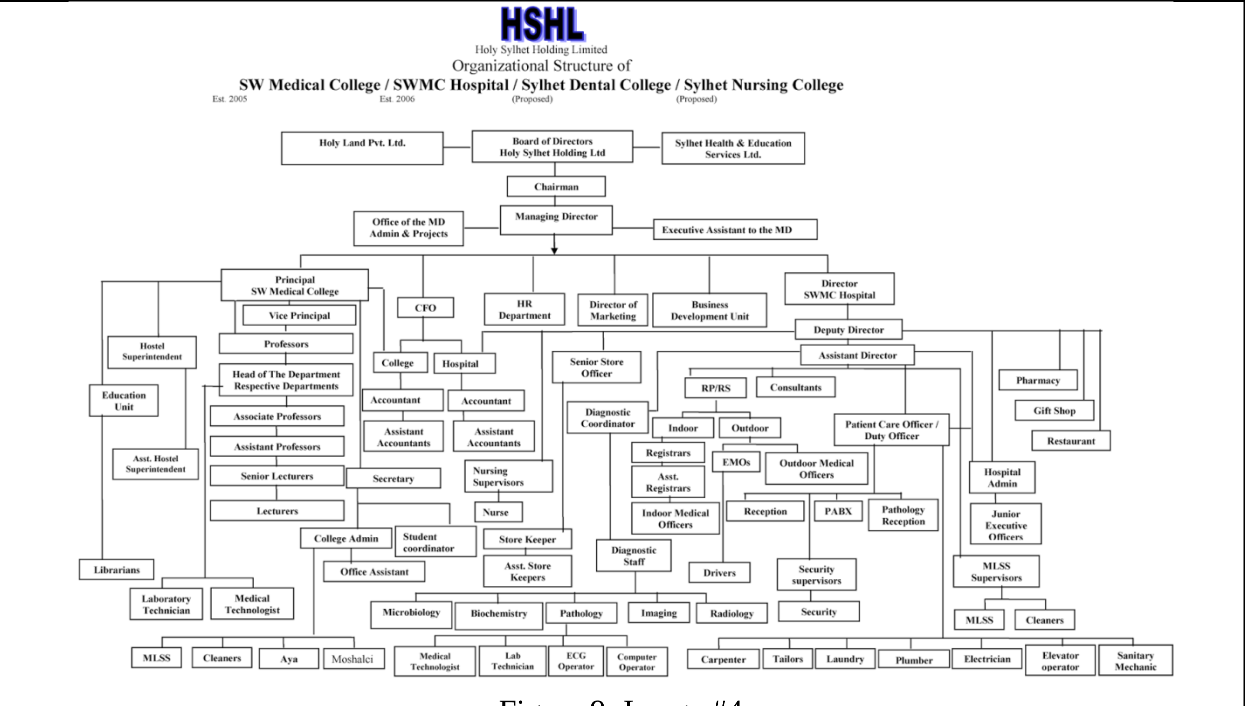


Figure 9: Image #4

Image #4 (Table B in Appendix A; Figure 9) includes two of the three defined characteristics of a CD. It has rectangles that cover a large portion of the image and are connected to other rectangles -- F01 and F07 are well within the st. dev. of the positive training set. The 3rd defined (rectangle dividers) characteristic (F09) falls well outside of the st. dev of the training set, and the ML could be expected to reject this image based on that. The distribution of the rectangles would also be expected to have the same effect, since two of the four sections are outside the st. dev. of the positive training set. It rates within the bounds of standard deviation for 15 of the 23 features extracted from positive images. If we cross-reference these features with our information gain analysis in Chart 1, we see that four of them are in the top ten with regards to information gain. These are rectangles' portion of image, rectangle size variation, rectangle connections and noise. The average deviation of these four from their respective features' average is only approximately 11%. The fact remains, that this image has great similarities (visually) with a CD and the ML might always have been expected to misjudge this image as a CD, without textual analysis. Of the 650 collected non-CDs, more than half of them had 2 of the 3 previously defined UML CD features in section 4.

8.2 Cross-validation and randomized tests

In order to test the feature extraction better, tests were implemented using Weka [14]. These tests might be a better indicator on the success of the feature extraction, but it is not possible to examine the false positives in this setting.

In addition to the chosen ML algorithm, SVM [15], five other algorithms were tested: Random forest [16]; J48 [17]; Logistic regression [18]; Simple cart [19]; and Decision table [20]. This was done to get a better view of success of the feature extraction, and to compare SVM with the other five algorithms.

Algorithm	**Overall**	**Negatives**	**Positives**
SVM	91.15%	92.90%	89.38%
Random forest	91.54%	92.77%	90.31%
J48	90.23%	90.31%	90.15%
Logistic regression	91.31%	91.38%	91.23%
Simple cart	91.00%	92.00%	90.00%
Decision table	90.54%	90.46%	90.62%

Table 3: Cross-validation test (10-fold)

The first of the two tests was a cross-validation test. That consists of the average of ten tests. The ML is trained with 90% of the image set and tested on the remaining 10% -- the image set is split into 10 sections and this is repeated for all sections. As can be seen in Table 3, the overall result for SVM was 0.90% less accurate than the average of the tests conducted using the classifier. But, the success in discriminating negative images was, though, 1.88% better than the average of classifier tests. Additionally, we can see that SVM does well in comparison with the other five ML. Only Random forest scores better overall, by 0.39%, but none of the other algorithms score better in discriminating negative images, which was considered an important constraint on the classifier. This test can be viewed as a good indicator of the success and accuracy of the feature extraction, because it tests the images in small portions, with a large training set each time, and covers many possible arrangements of the data set.

Algorithm	**Overall**	**Negatives**	**Positives**
SVM	91.54%	92.97%	90.15%
Random forest	91.92%	92.19%	91.77%
J48	88.65%	90.23%	87.12%
Logistic regression	91.15%	91.80%	90.53%
Simple cart	89.23%	90.23%	88.26%
Decision table	89.42%	88.77%	90.15%

Table 4: 60/40 randomized test

The second test was done by randomizing the images and then training and testing -- similar to the tests that were conducted on the classifier. The main randomized test, that included the other five ML, also had 60/40 split for training/testing. As we can see in Table 4, the accuracy of SVM increases a little from the cross-validation test (Table 2). Additionally, the SVM stays above the other ML's in relation to success in classifying negative images -- as it does in the cross-validation test.

When the results of the randomized Weka tests are compared to the average of the classification tests, it can be seen that the overall accuracy is 0.51% less accurate in the Weka tests. The accuracy on negative images is 1.95% better in the Weka tests though. This can be viewed as a positive indicator, since the original goal was 95% accuracy on negative images -- it misses that mark by 2.03%, as opposed to 3.98% in the classification tests.

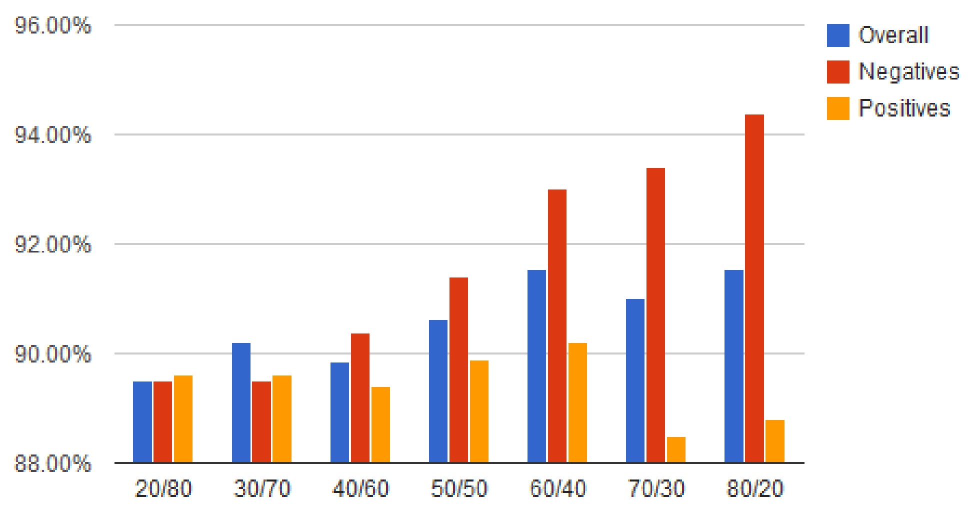


Chart 2: SVM Randomized tests

To get a better overview of the SVM, seven different randomized tests were implemented in Weka. The result of these tests can be seen in Chart 2, which consists of various training/testing splits. What is apparent in the tests is that when the size of the training set is increased, the success ratio increases. Specifically, it can be seen that in the 80/20 test, the success on negative images is the highest. This corresponds well to the goals of the classifier: having high overall success rate, but focusing more on having high success rate in negative images. These findings strongly indicate that, to train the ML with all of the 1300 images before usage would increase the accuracy.

8 Results

RQ0:

On a basic level, there are four steps to automating the classification of UML CDs. The first step is image processing, where relevant preliminary lines and shapes are extracted from the image. The second step is processing the extracted lines and shapes, resulting in descriptive features. The third step is training, where your machine learner of choice is trained, using the data you've extracted from a set of relevant images. Lastly, you must conduct testing to ascertain the accuracy of your classifier. If the result are adequate, success has been achieved.

RQ1:

Chart 1, in section 5.1, demonstrates the information gain of each of the extracted features, where each feature is ranked between 0 and 1 -- the closer the feature is to 1, the more influence it has. As was discussed in the previous section, eliminating all of the four features that score 0 in the chart, did not increase the success of the classifier. To test this further, 4 tests were implemented to test the effect of removing each of the features. The tests were 10-fold cross-validation tests. This can be seen in Chart 3, along with the test results with all features.

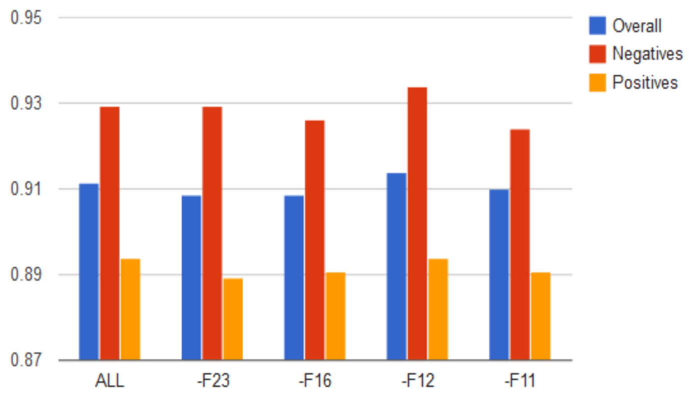


Chart 3: Cross validation of elimination tests

All of the features did have positive effects on the outcome, with the exception being F12. By removing F12, the overall success percentage increased from 91.15% to 91.38%, and it scored better in classifying negative images by 0.47%. As a result, F12 was removed from the feature extraction.

RQ2:

The chosen algorithm, SVM, scored highest in classifying negative images. Table 2, in the previous section, demonstrates the comparison of five different algorithms, through 10-fold cross-validation tests. SVM scored second overall, 0.39% behind Random Forest -- but 0.13% better in negative images. Additionally, in the 60/40 randomized tests (Table 4, section 7), the SVM scored 0.65% better than Random Forest on negative images, but the same in overall accuracy. Since negative accuracy was considered more important than positive accuracy, the SVM was considered a good fit.

RQ3:

There are three accuracy measurements to be considered when assessing the results; overall; negatives; and positives. As previously stated, the most important of these is negative accuracy. Positives' accuracy must of course be reasonably high, but since the lowest such accuracy number throughout the testing is 86.15%, the negative accuracy becomes the sole criteria for accuracy assessment. The highest recorded positives' accuracy tested, was 95% in Test A, table 2. However, tests B and C, where only the training/testing images are randomly changed, returned considerably worse results with regard to negative accuracy, lowering the average of the tests down to 91.02%. For the most reliable test results, we look to the 10-fold cross-validation test. Using that technique maximises the potential of the image database, effectively giving the largest possible training set, as well as the largest possible testing set. Based on that test, the classifier reached a negative accuracy of 92.90%.

RQ4:

When examining the top 3, 6, and 10 features (from Chart 1), in Chart 4, it is apparent that when features are added, the success ratio raises. Specifically, the negative success raises in proportion to the overall success rate -- aligned with the success-rate goals of this research.

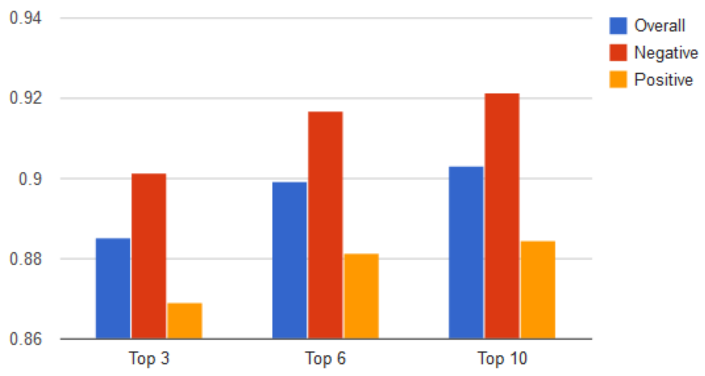


Chart 4: Set-test, SVM cross-validation tests

9 Discussion

The main limitation of the classifier is content comprehension. Diagrams that have the same visual characteristics as CDs are likely to be returned as positives. The time-constraints on this research did not allow for investigation and implementation of textual analysis, which could have prevented many of the false-positives that might occur. That would likely increase the success rate of the classifier.

There were two main fields that this paper covered: image processing and feature extraction; and machine learning. The former involved various issues that had be addressed in an effort to successfully implement the classifier. Mainly, the problem had to do with an external algorithm that was used to extract lines from the image -- Hough transformation. The problem lay in the output of the algorithm. It missed line segments and did not return all of the necessary lines to process the image correctly. This was addressed by joining Suzuki85 with the algorithm and joining together lines that represented the same straight lines. Based on various tests on machine learners we can see that the SVM was a good fit for this classifier and that, the larger the training set is, the more accurate the classifier becomes.

In the beginning of this research, it was considered a priority to eliminate false-positives, even if it meant increasing the number of false-negatives. The classifier was intended for the gathering of images depicting UML CDs into a database, and false-negatives don't compromise the integrity of said database, as false-positives do. In the initial planning phase, the desired outcome would have at least a total accuracy of 90%, minimum 95% accuracy on negative images, and minimum 85% accuracy on positive images. The authors may have been a bit optimistic about the accuracy on negative images, when the textual analysis isn't included in the research. Without that kind of analysis, diagrams that have the same features as CDs will always be difficult to distinguish. As a result, the cross-validation results show a 92.90% accuracy on negative images, and the average accuracy in the classification tests resulted in 92.05% accuracy on negative images -- missing the goal by 2.10% and 2.95%, respectively. On the positive side, that is not far from the goal, and by adding the text analysis to the classifier we can expect the success rate to go well beyond the desired outcome.

There are two things that can be considered when we think about future work in relation to this classifier: increased success rate by adding textual analysis; and, classifications of different types of diagrams. These two are not mutually exclusive. To classify different, but similar, diagrams, new features have to be researched, and the textual analysis would help in classifying those diagrams also. These various types of diagrams can include, but not limited to: UML Sequence diagrams; UML State machines; and UML Use cases.

10 Conclusion

In this paper we propose a classifier of UML Class diagrams. This classifier can be of interest to academia, who focus on researching CDs. We demonstrate what features in an image help classify it as a CD, and how to collect these features through image processing. 1300 images were collected for testing, with them equally divided between CDs and non-CDs. The features are then tested with -- in addition to the chosen SVM -- five Machine learners, to compare the success of the classification. Cross-validation testing on the MLs results in SVM being 2nd (0.39 less) in overall accuracy, but 1st in success on negative (0.13 more) images -- with a 91.15% overall success rate. Randomized 60/40 tests also return SVM second -- 0.38 less in overall then Random forest, but 0.78 better negative success -- with 91.54% correct classifications.

11 Acknowledgement

Special thanks are given to our supervisors, Prof. Michel Chaudron and Truong Ho Quang, for guidance and help throughout the lifetime of this research. Additionally, we would like to thank Mohd Hafeez Osman for assistance in relation to machine learners.

12 References

1. B. Karasneh & M. R.V. Chaudron, *Extracting UML Models from Images*, 39th Euromicro Conference Series on Software Engineering and Advanced Applications, 2013.
2. E. Lank, J. Thorley & S. Chen, *An interactive system for recognizing hand-drawn UML diagrams*, Proceedings of the IBM Center for Advanced Studies Conference, CASCON 2000, Mississauga, Ontario, Canada, November 2000, pp. 1 - 15.
3. B. T. Messmer & H. Bunke, *Automatic Learning and Recognition of Graphical Symbols in Engineering Drawings*, First International Workshop University Park, PA, USA, August 1995, pp. 123-134
4. L. Fu & L. B. Kara, *From engineering diagrams to engineering models: Visual recognition and applications*, Computer-Aided Design, Volume 43, Issue 3, March 2011, pp. 278–292

5. R. O. Duda & P. E. Hart, *Use of Hough transformation to detect lines and curves in pictures*, Magazine Communications of the ACM, Volume 15, Issue 1, January 1972, pp. 11-15

6. J. Canny, *A Computational Approach to Edge Detection*, Pattern Analysis and Machine Intelligence, IEEE Transactions, Volume PAMI-8, Issue 6, November 1986, pp. 679-698

7. S. Suzuki & K. Abe, *Topological Structural Analysis of Digitized Binary Images by Border Following*, Computer Vision Graphics and Image Processing, Volume 30, Issue 1, April 1985, pp. 32–46

8. B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A training algorithm for optimal margin classifiers*, Proceedings of the fifth annual workshop on Computational learning theory, ACM, New York, 1992

9. A. Rosenfeld & A. C. Kak, *Digital Picture Processing*, 2nd edition, Vol. 2, Academic Press, New York, 1982

10. U. Ramer, *An iterative procedure for the polygonal approximation of plane curves*, Computer Graphics and Image Processing, Volume 1, Issue 3, November 1972, pp. 244–256

11. OpenCV - http://www.opencv.org

12. J. Rumbaugh, I. Jacobson and G. Booch, *UML Proposal to the Object Management Group*, version 1.1, ad/97-08-02 (1 September 1997)

13. Magick++ - http://www.imagemagick.org/Magick++/

14. Weka 3: Data Mining Software in Java - http://www.cs.waikato.ac.nz/ml/weka/

15. J. C. Platt, *Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods*, Microsoft Research, March 1999

16. L. Breiman, *Random Forests*, Machine Learning, October 2001, Volume 45, Issue 1, pp. 5-32

17. J. R. Quinlan, *C4.5: Programs for Machine Learning*, 1993, Morgan Kaufmann Publishers, San Mateo, CA

18. S. L. Cessie, J. C. V. Houwelingen, *Ridge estimators in logistic regression,* Journal of the Royal Statistical Society, Vol. 41, No. 1, 1992

19. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, 1984, Wadsworth International Group, Belmont, California

20. R. Kohavi, *The Power of Decision Tables*, Machine Learning: ECML-95 Lecture Notes in Computer Science, Volume 912, 1995, pp. 174-189

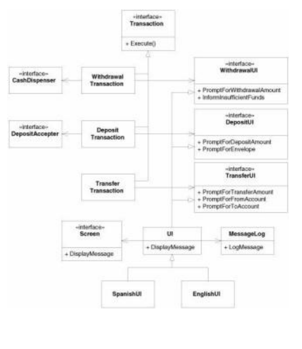
Appendix A:

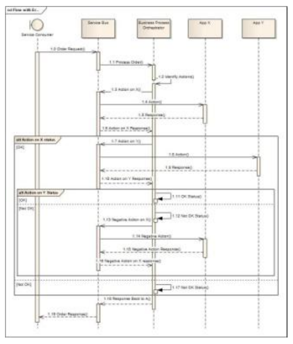
Features	**Average**	**Std. dev.**
F01	0.306	0.124
F02	0.611	0.321
F03	0.262	0.092
F04	0.260	0.094
F05	0.231	0.093
F06	0.247	0.100
F07	0.755	0.192
F08	0.213	0.209
F09	0.747	0.213
F10	0.040	0.088
F11	0.032	0.073
F12	0.026	0.089
F13	0.246	0.075
F14	0.322	0.128
F15	0.0001	0.002
F16	0.008	0.025
F17	0.795	0.347
F18	0.00003	0.0002
F19	0.017	0.178
F20	0.203	0.127
F21	0.733	0.165
F22	0.067	0.068
F23	0.018	0.020

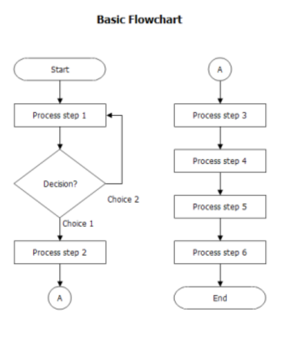
Table A: Test A, positive training set

Features	**Image #1**	**Image #2**	**Image #3**	**Image #4**	**Image #5**
F01	0.381	0.496	0.305	0.274	0.339
F02	0.749	1.131	1.702	0.641	0.362
F03	0.186	0.208	0.361	0.101	0.324
F04	0.273	0.296	0.162	0.423	0.356
F05	0.249	0.034	0.332	0.054	0.236
F06	0.291	0.462	0.146	0.422	0.084
F07	0.615	0.9	0.923	0.777	0.667
F08	0.846	0.5	0.538	0.979	0.5
F09	0.153	0.4	0.462	0.021	0.5
F10	0	0.1	0	0	0
F11	0.077	0.2	0	0.128	0
F12	0.462	0.4	0	0.085	0
F13	0.303	0.342	0.330	0.172	0.385
F14	0.364	0.317	0.323	0.254	0.312
F15	0	0.144	0	0	0
F16	0.02	0.134	0.065	0.011	0
F17	0.848	2.066	0.435	0.802	0.464
F18	0	0	0	0.0002	0
F19	0	0	0	0	0
F20	0.18	0.368	0.393	0.265	0.096
F21	0.804	0.713	0.741	0.922	0.957
F22	0.067	0.079	0.025	0.053	0.014
F23	0.003	0.032	0.01	0.002	0.0009

Table B: false-positives in test set







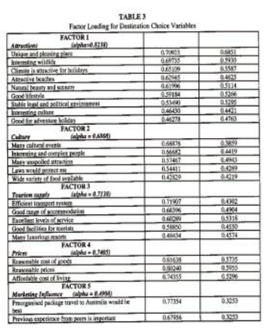
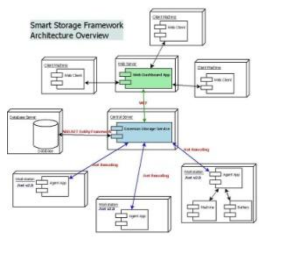
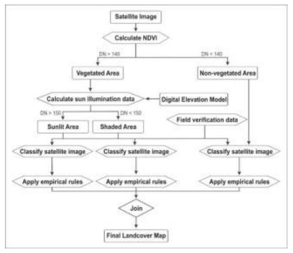
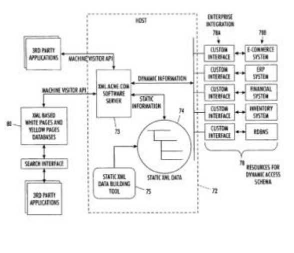


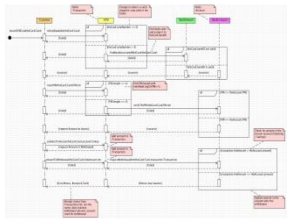
TABLE 3
Factor Loading for Destination Choice Variables

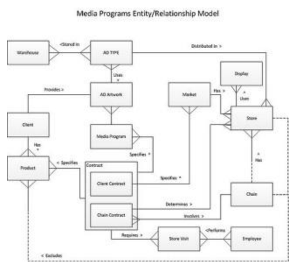
FACTOR 1 *Attractions* (*alpha=0.8238*)		
Unique and pleasing place	0.70923	0.6851
Interesting wildlife	0.69735	0.5930
Climate is attractive for holidays	0.65109	0.5587
Attractive beaches	0.62945	0.4625
Natural beauty and scenery	0.61996	0.5114
Good lifestyle	0.59184	0.5266
Stable legal and political environment	0.53490	0.5295
Interesting culture	0.46430	0.4421
Good for adventure holiday	0.46278	0.4763
FACTOR 2 *Culture* (*alpha = 0.6808*)		
Many cultural events	0.68876	0.3859
Interesting and complex people	0.66682	0.4419
Many unspoiled attraction	0.57467	0.4943
Laws would protect me	0.54411	0.4289
Wide variety of food available	0.42829	0.4219
FACTOR 3 *Tourism supply* (*alpha = 0.7138*)		
Efficient transport system	0.71907	0.4302
Good range of accommodation	0.68396	0.4904
Excellent levels of service	0.60289	0.5318
Good facilities for tourists	0.58850	0.4550
Many luxurious resorts	0.48434	0.4574
FACTOR 4 *Prices* (*alpha = 0.7405*)		
Reasonable cost of goods	0.81638	0.5735
Reasonable prices	0.80240	0.5955
Affordable cost of living	0.74355	0.5296
FACTOR 5 *Marketing Influence* (*alpha = 0.4998*)		
Preorganised package travel to Australia would be best	0.77354	0.3253
Previous experience from peers is important	0.67956	0.3253

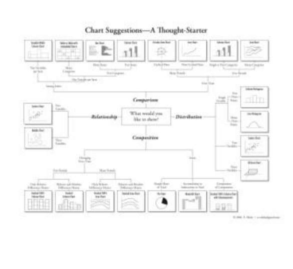


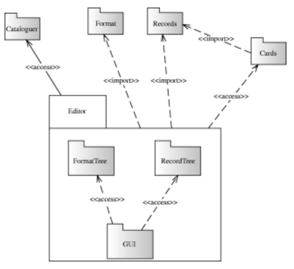


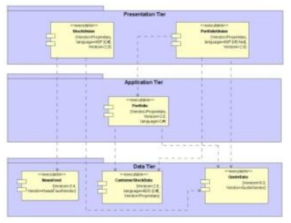




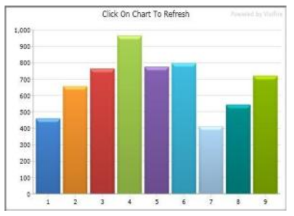








God
The Muslim Brotherhood
Freedom and Justice Party
The Egyptian Electorate
1952 Officer's Coup
Gen Hussein Tantawi, Chairman
Gen Hussein Tantawi, Minister of Defence
Supreme Council of the Armed Forces
President Muhammad Mursi
Prime Minister
Cabinet
Flow Chart of Authority in Today's Egypt



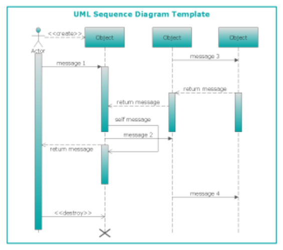
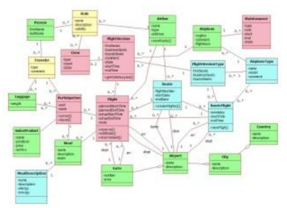
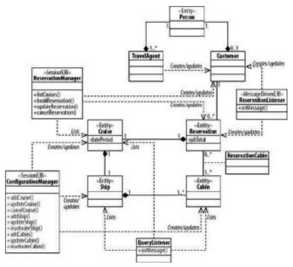
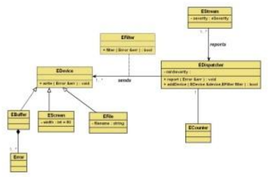
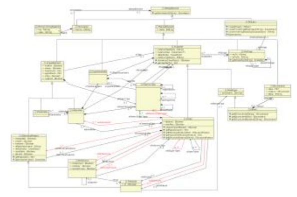
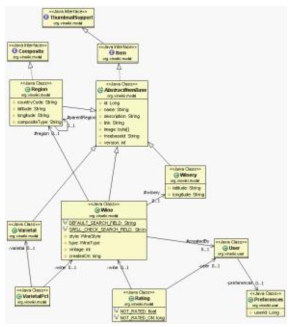
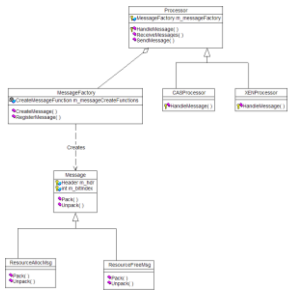
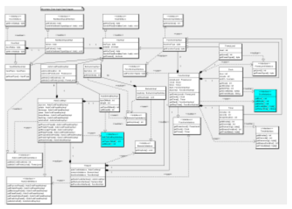
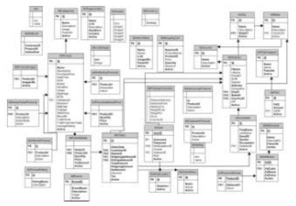
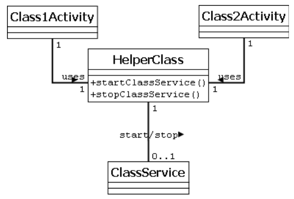
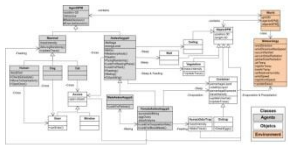
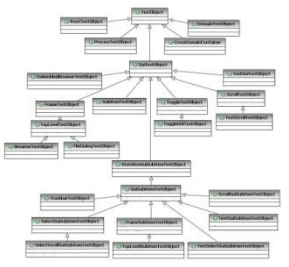
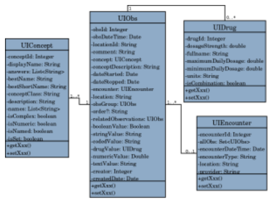
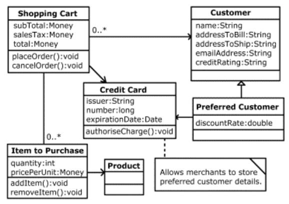


Table C: Sample of negative images

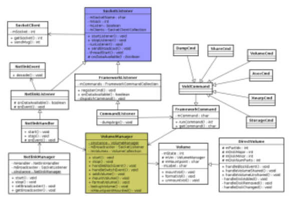
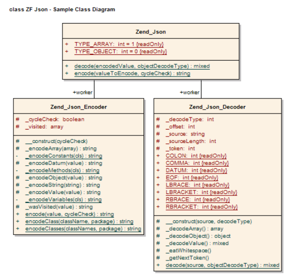



Table D: Sample of positive images

Appendix B:

Ellipse equation:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1$$

a = width, b = height