**CHAPTER THREE**

**MATERIALS AND METHODS**

**3.1 Methodology**

The Methodology for the Proposed System Design is Structured Analysis and Design Technique (SADT). SADT is a structured analysis modeling language, which uses two types of diagrams: activity models and data models. It was developed in the late 1960s by Douglas T. Ross, and was formalized and published as IDEF0 in 1981.

**3.1.1 Choice and Justification of the Methodology**

SADT has proven to be successful in the development of software systems, specifically in the requirements gathering phase. SADT notations consist of box-arrow diagrams (blocks), with four arrows on each side defined as: input, output, control and mechanism and one activity in the middle as shown in figure 3.1. Their definitions consist of the following:

1. **Activity:** An activity is any function or process that serves to transform inputs into outputs
2. **Input:** The data/information required by an activity to start the transformation process
3. **Output:** the data/information produced by the activity as a result of this transformation
4. **Control:** Any constraint that affects the behavior of activity in some way
5. **Mechanism:** Persons, resources, or any means that are required to run the activity

Systems Analysis and Design Technique (SADT) is an exciting, active field in which analysts continually learn new techniques and approaches to develop systems more effectively and efficiently.

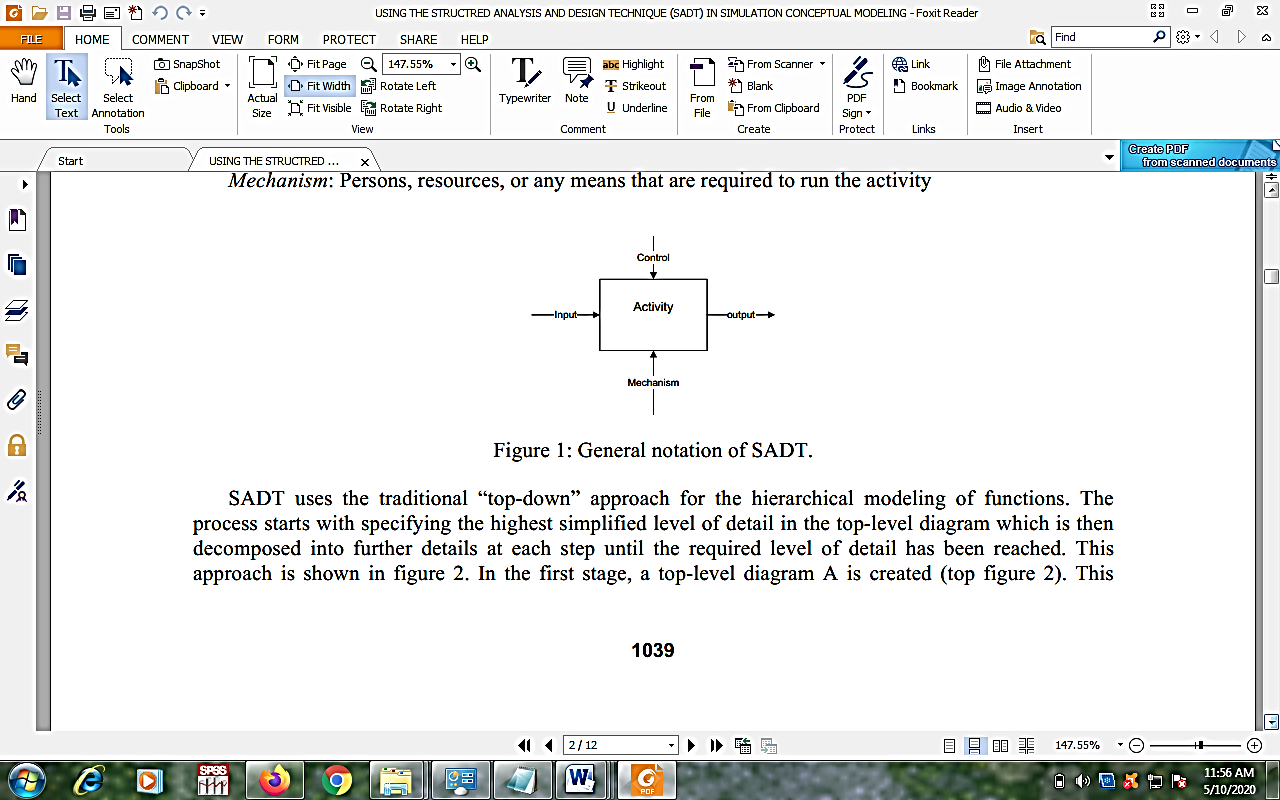


Figure 3.1: General Notation of SADT Methodology (**Source:** Fahim, 2014)

However, there is a core set of skills that all analysts need to know no matter what approach or methodology is used. All information systems projects move through the four phases of planning, analysis, design, and implementation; all projects require analysts to gather requirements, model the business needs, and create blueprints for how the system should be built; and all projects require an understanding of organizational behavior concepts like change management and team building. The major goal of systems analysis and design is to improve organizational systems.

**3.2 Analysis of the Existing System**

**Analysis of the Existing System**

The existing system is a model for Piracy Detection and Prevention using SIFT based on Earth Mover’s Distance (EMD). The model was designed as a robust system with a centralized activation server which is capable enough to make secured validations using image matching and features extraction based on Earth movers distance algorithm and SIFT algorithms. The research work discussed about a robust yet efficient method for avoiding software piracy. In their work, they first elaborated on the fact that whenever someone buys software, the software publisher delivers software and its associated license to the customer. A complete usage policy and license agreement is clearly mentioned in software license documents. In other words, the customer is actually purchasing license to use that software. Breaking those terms and policies specified in license, some people redistribute this software by creating copies of it and remove any licenses associated with that respective software, resulting in software piracy. This pirated software is similar to the original software except for its license. Using pirated software without a license is against law. Despite piracy being considered a felony‟, people still lean on using pirated software. This is an analogy to theft in real life, just as theft cannot be entirely prevented, we cannot stop people from using pirated software. With increasing usage of broadband internet, peer-to-peer and torrent sharing techniques, piracy of software is becoming easier. This mass attitude developed by pirates among people leads to drastic increase in piracy due to which is almost one third of software content available today is pirated. The researchers stated that one of the primary reasons for most of the software being pirated is the fact that software sales can be monitored and recorded while software usage cannot be monitored, this enables unethical software owners or pirates to distribute these programs or copy these copyrighted content using many tools, media and sharing directories. Many underground Forums and discussion groups are filled with copies of copyrighted software; even a novice software user can login to these systems and can download software, free of cost.

The existing system is a robust system with a centralized activation server is designed which is capable enough to make secured validations using image matching and features extraction based on Earth movers distance algorithm and SIFT algorithms. Software producer integrates an activation tool along with software distribution which will be used to perform validations and activation at user end. Whenever a buyer purchases software, buyer will submit his image, this image is stored in database and used for the purpose of matching, features are extracted from this image using SIFT and stored in database[19]. During software activation using integrated tool, user submits the same image which he had submitted during software purchase, or takes a photo using system webcam. This image is sent to activation server along with user system properties. The resulted image is then compared with image in database using earth mover distance algorithm after verification and validation, a unique key of 2MB size is generated based on features and system properties along with a tracking key to track user software activation process and usage. This key is shared with user and key file is stored in user system, and software is now activated successfully.

Face Detection and Feature Extraction Face recognition can be applied for a wide variety of problems like image and film processing, human-computer interaction, criminal identification etc. This has motivated the researchers to develop computational models to identify the faces, which are relatively simple and easy to implement.

Scale Invariant Feature Transform (SIFT) features are features extracted from images to help in reliable matching between different views of the same object [18]. The extracted features are invariant to scale and orientation and are highly distinctive of the image. They are extracted in four steps. The first step computes the locations of potential interest points in the image by detecting the maxima and minima of a set of Difference of Gaussian (DoG) filters, applied at different scales all over the image. Where DoG image 𝐷(𝑥, 𝑦, 𝝈) is given by 𝐷 𝑥, 𝑦, 𝜎 = 𝐿 𝑥, 𝑦, 𝑘𝑖𝜎 − 𝐿 𝑥, 𝑦, 𝑘𝑖𝜎 where 𝐿(𝑥, 𝑦,𝐾𝜎)is convolution of the original image 𝐼(𝑥, 𝑦) with the Gaussian blur 𝐺(𝑥, 𝑦, 𝑘𝜎) at scale k𝝈 that is 𝐿 𝑥, 𝑦, 𝑘𝜎 = 𝐺 𝑥, 𝑦, 𝑘𝑖𝜎 ∗ 𝐼 𝑥, 𝑦 Then, these locations are refined by discarding points of low contrast. An orientation is then assigned to each key point based on local image features. Finally, a local feature descriptor is computed at each key point. This descriptor is based on the local image gradient that is transformed according to the orientation of the key point to provide orientation in variance. Every feature is a vector of dimension 128, distinctively identifying the neighborhood around the key point.

SIFT approach is followed and features are extracted for user submitted image, because using SIFT approach we can easily compare images despite their orientation in different directions. These generated features are used for image comparison and the generation of unique key for validation. This system is built in two phases. In the first part, image features are generated from user submitted image during purchase phase. Generated image features are added to database along with software ID. Second part is building a software installer that is used for activation of software this is build and distributed along with software, this installer connects to activation server and performs activation process

**Figure 3.2:** Architecture of the Existing System

The existing system involves three participants namely: the buyer, vendor and software manufacturer. The activities between the vendor and software manufacturer were broadly explained with less activity from the buyer. The existing system has two main phases:

1. Purchase Phase
2. Software Activation/Installation Phase

**Purchase Phase**

The purchase phase is where the vendor purchases the software from the software manufacturer and the buyer from the vendor. Here the vendor is expected to submit his/her image and the image features are generated from the vendor submitted image.

**Purchase Phase Activities (Software Vendor)**

The Purchase Phase activities between the vendor and the software manufacturer are as follows:

**Step 1:** Request for the purchase of the software from the software manufacturer

**Step 2:** Runs activation wizard during installation

**Step 3:** Activation tool asks the vendor for his/her image for purpose of validation during activation and hence vendor submits the respective image.

**Step 4:** Image is stored in a database alongside the software ID.

**Software Activation/Installation Phase**

The software activation/installation phase involves the actual activation of the purchased software with the aid of the authentication key and tracking key. The software activation/installation phase works together with the activation centralized server.

**Activation/installation Phase Activities**

The activities involved in the activation/installation phase are carried out between the activation tool and the activation server. These activities are as follows:

1. Generated image features are added to database along with software ID.
2. Validation tool communicates with activation server and server performs activation operation.
3. Vendor submits the same image which he had submitted during software purchase, or takes a photo using system webcam.
4. Image is sent to activation server along with user system properties.
5. Server verifies and validates the image by comparing it with the image in the database earlier submitted by the vendor during the purchase phase
6. After the verification and validation process, a unique key (authentication key) of 2MB size is generated based on features and system properties which is used for validation during product activation along with a tracking key to track user software activation process and usage.
7. The unique key (authentication key) and tracking key is sent to the validation tool (shared with vendor) and the key file is stored in the vendor system
8. Vendor successfully activates the software

**Centralized storage Activities**

1. Receive image and software ID and extract features using SIFT F1=SIFT(INPIMG);
2. Fetch initial features calculated using user submitted image during purchase, from database using Software ID F2=SIFT(DBIMG);
3. Generate signatures S1 and S2 using F1 and F2 and calculate difference using EMD e = emd(&s1, &s2, dist, 0, 0);
4. Generate validation and authentication keys based on image features and Software ID and send it to User.

**Disadvantage of the existing system**

The following disadvantages of the Existing System are:

1. The existing system, did not give full details of the activities of the buyer which refers to the software buyer who most times purchases his /her software from the vendor. There is a need for the buyer to complete the verification and validation process for any software he/she purchase to be sure that the software is genuine and not pirated.
2. The validation and verification process during the purchase phase did not request for the finger print of the vendor. A combination of both image and finger print authentication of the user would tightened the security measures put in place by the software providers to avoid software piracy.
3. The existing system is not fully efficient due to the lack of provision of a branch count value. A branch count value will enable software providers to determine if the software code has been tampered. With branch count values assigned to software just like authentication key is assigned to software products, it will help boost the security measures to detect software piracy.

**3.3 Analysis of the Proposed System**

The proposed system is an improved detection and prevention system for online software piracy using hybridized algorithms. It involve four techniques using, product name, license key and registration details of the user alongside user image and fingerprint and extraction of the hardware details of the user system. This will prevent the unauthorized sharing of the software with users.

**Vendor/Buyer Activity**

1. Initialize the system
2. Login to application
3. Request for download of Software
4. User now runs activation wizard during software installation
5. Activation tool request user to fill user registration Form
6. Activation tool request user for its image for the purpose of validation and hence user submits the respective image
7. Activation tool request user for its fingerprint for the purpose of validation and hence user submits the respective fingerprint

**Centralized Server**

1. Receive image, fingerprint, software ID and extract features
2. Fetch initial features Generate signatures
3. Generate authentication keys, branch count value based on image, fingerprint features and Software ID and send it to User.

**3.6 Algorithm of the Proposed System**

In order to choose an effective and efficient algorithm to develop the proposed system, a hybridized algorithm made up machine learning classification algorithms were researched and evaluated for this research. These machine learning classification algorithms include:

1. Random Forest
2. Naive Bayes
3. Support Vector Machine

These algorithms were for the design of the model for this research.

**3.6.1 Random Forest Algorithm**

Random Forest randomly selects observations, builds a decision tree and then the result is obtained based on majority voting. No formulas are required here. One of the main advantages is that it reduces the risk of over fitting and the required training time. Additionally, it offers a high level of accuracy. Random Forest algorithm runs efficiently in large databases and produces highly accurate predictions by estimating missing data.

**Steps for Random for Algorithm**

**Step 1: Split the dataset into subsets**

A random forest is an ensemble of decision trees. To create many decision trees, we must divide the dataset we have into subsets.

**Step 2: Train decision trees** of the subsets

After splitting the dataset into subsets, we train decision trees on these subsets. The process of training is the same as it would be for training an individual tree

**Step 3:** Aggregate the results

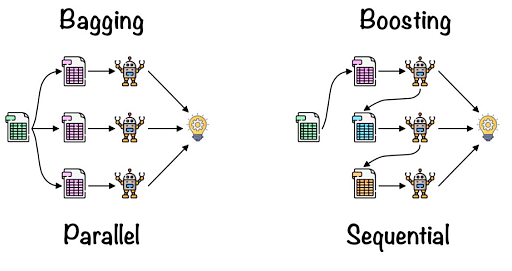
Each individual tree contains one result that depends on the tree’s initial data. To get rid of the dependence on the initial data and produce a more accurate estimation, we combine their output into one result. Different methods of aggregating the results can be used. For example, in the case of classification, voting by performance is used quite often. The classification method is used to determine which “class” a target variable is most likely to belong to.

**Step 4:** Validate the model

After we complete the training procedure with the training data and run the tests with the test dataset, we perform the hold-out validation procedure

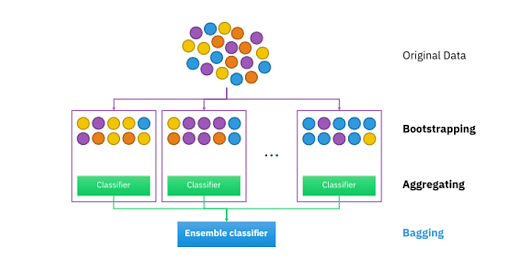
This combination of multiple models is called Ensemble. Ensemble uses two methods:

1. Bagging: Creating a different training subset from sample training data with replacement is called Bagging. The final output is based on majority voting.
2. Boosting: Combing weak learners into strong learners by creating sequential models such that the final model has the highest accuracy is called Boosting. Example: ADA BOOST, XG BOOST.



**Figure 3.8:** Ensemble Methods

**Bagging**: From the principle mentioned above, Random forest uses the Bagging code. Bagging is also known as Bootstrap Aggregation used by random forest. The process begins with any original random data. After arranging, it is organized into samples known as Bootstrap Sample. This process is known as Bootstrapping. Further, the models are trained individually, yielding different results known as Aggregation. In the last step, all the results are combined, and the generated output is based on majority voting. This step is known as Bagging and is done using an Ensemble Classifier.

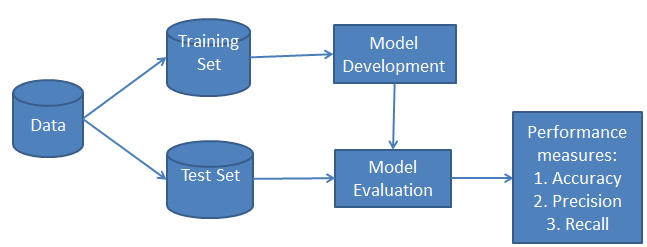


**Figure 3.8:** Ensemble Classifier

**3.6.2 Naïve Bayes Algorithm**

The Naïve Bayes classifier is a popular supervised machine learning algorithm used for classification tasks. It belongs to the family of generative learning algorithms, which means that it models the distribution of inputs for a given class or category. This approach is based on the assumption that the features of the input data are conditionally independent given the class, allowing the algorithm to make predictions quickly and accurately. Meaning, it is a classifier that assumes the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes classifiers are considered as simple probabilistic classifiers that apply Bayes’ theorem. This theorem is based on the probability of a hypothesis, given the data and some prior knowledge. The naive Bayes classifier assumes that all features in the input data are independent of each other, which is often not true in real-world scenarios. It learns the probability of every object, its features, and which groups they belong to. For example, you cannot identify a car based on its features and color as there are many cars with similar attributes. But, you make a probabilistic prediction about the same, and that is where the Naive Bayes Algorithm comes in.

The classification has two phases, a learning phase and the evaluation phase. In the learning phase, the classifier trains its model on a given dataset, and in the evaluation phase, it tests the classifier's performance. Performance is evaluated on the basis of various parameters such as accuracy, error, precision, and recall.



## Probability, Bayes Theory, and Conditional Probability

### Probability is the base for the Naive Bayes algorithm. This algorithm is built based on the probability results that it can offer for unsolvable problems with the help of prediction.

### Probability

Probability helps to predict an event's occurrence out of all the potential outcomes. The mathematical equation for probability is as follows:

Probability of an event = Number of favorable event

Total Number of outcomes

0 < = probability of an event < = 1. The favorable outcome denotes the event that results from the probability. Probability is always between 0 and 1, where 0 means no probability of it happening, and 1 means the success rate of that event is likely.

**Bayes Theory**

Bayes Theory works on coming to a hypothesis (H) from a given set of evidence (E). It relates to two things: the probability of the hypothesis before the evidence P(H) and the probability after the evidence P(H|E). The Bayes Theory is explained by the following equation:

P (H|E) = (P (E|H) \* P (H))

P(E)

In the above equation,

1. P(H|E) denotes how event H happens when event E takes place.
2. P(E|H) represents how often event E happens when event H takes place first.
3. P(H) represents the probability of event X happening on its own.
4. P(E) represents the probability of event Y happening on its own.

The Bayes Rule is a method for determining P(H|E) from P (E|H). It provides a way of calculating the probability of a hypothesis with the provided evidence.

### Conditional Probability

Conditional probability is a subset of probability. It reduces the probability of becoming dependent on a single event. You can compute the conditional probability for two or more occurrences. When you take events X and Y, the conditional probability of event Y is defined as the probability that the event occurs when event X is already over. It is written as P(Y|X). The mathematical formula for this is as follows:

P (Y|A) = P(X and Y)

P(X)

**Bayesian Probability**

Bayesian Probability allows calculating the conditional probabilities. It enables to use of partial knowledge for calculating the probability of the occurrence of a specific event. This algorithm is used for developing models for prediction and classification problems like Naive Bayes.

**Gaussian Naive Bayes algorithm**

**There are different types of Naïve Bayes algorithms namely:** Gaussian Naive Bayes, Optimal Naive Bayes, Bernoulli Naive Bayes and Multinomial Naive Bayes. This research work adapted the Gaussian Naive Bayes is a straightforward algorithm used when the attributes are continuous. The attributes present in the data should follow the rule of Gaussian distribution or normal distribution. it is used to calculate the to calculate the mean and standard deviation for the training data.

**Steps for Naïve Bayes Algorithms**

**Step 1: Import basic libraries**

**Step 2: Importing the dataset**

**Step 3: Data preprocessing**

**Step 4: Training the model**

**Step 5: Testing and evaluation of the model**

**Step 6: Visualizing the model**

**3.6.3 Support Vector Machine**

The idea behind the SVM classifier is to come up with a hyper-lane in an N-dimensional space that divides the data points belonging to different classes. However, this hyper-pane is chosen based on margin as the hyper plane providing the maximum margin between the two classes is considered. These margins are calculated using data points known as Support Vectors. Support Vectors are those data points that are near to the hyper-plane and help in orienting it.

**Steps for Naïve Bayes Algorithms**

**Step 1:** Load the important libraries

**Step 2:** Import dataset

**Step 3:** Divide the dataset into train and test

**Step 4:** Fitting the SVM classifier model

**Step 5:** Coming up with predictions

**Step 6:** Evaluating model’s performance

**3.6.4 Neural Network**

**3.6.5 Hybridized Algorithm**

**Step 1:** Start

**Step 2:** Upload the dataset of code from Kaggle

**Step 3:** Train the dataset using neural network and python programming language

**Step 4:** Initialize the system

**Step 5:** Register with the application

**Step 6:** Login to the application

**Step 7:** Software Piracy Analysis

1. Enter into Software Detection Environment Module
2. Request for Software Details
3. Submit Software Details

**Step 8:** Analyze and detect if software is genuine or pirated.

**Step 9:** Display Software piracy analysis result

**Step 10:** Store details of all software analyzed

**Step 11:** Store details of the sources of all software analyzed.

Neural Network will be used to train the datasets that will used for implementing the proposed system will be trained

**3.7 Design of the Proposed System**

**3.8 Dataset**

A data set is a structured collection of data points related to a particular subject. Data sets can be tabular or non-tabular. Tabular data sets contain structured data that is organized by rows and columns. Non-tabular data sets contain unstructured data contained by brackets. The dataset used for this research was source from Kaggle Repository

**3.9 Use Case Activity Diagram of the Proposed System Design**

**Start**

**Initialize the system**

**Register with the application**

**Login to the application**

**View software detected results**

**Submit details of Software**

**Log out of the application**

**Provide software details for software detection environment**

3.10 Class Diagram of the Proposed System.

3.11 Sequence Diagram of the Proposed System

3.12 High Level Diagram of the Proposed System

3.13 Methodology of the Proposed System

3.14 Justification of the Design Methodology Used

3.15 Datasets

3.16 Advantages of the Proposed System

3.17 Choice and Justification of Programming Language for the Proposed System