**ASSIGNMENT NO.**

**Aim**: Modelling, Designing and Implementing Binary search.

**Objective:** Implement binary search using Divide and Conquer Strategies and model it using object-oriented software design technique.

**Theory**:

· Binary Search :

A binary search or half**-**interval search [algorithm](https://en.wikipedia.org/wiki/Algorithm) finds the position of a target value within a sorted HYPERLINK "https://en.wikipedia.org/wiki/Sorted\_array" HYPERLINK "https://en.wikipedia.org/wiki/Sorted\_array"array.The binary search algorithm begins by comparing the target value to the value of the middle element of the sorted array. If the target value is equal to the middle element's value, then the position is returned and the search is finished. If the target value is less than the middle element's value, then the search continues on the lower half of the array; or if the target value is greater than the middle element's value, then the search continues on the upper half of the array. This process continues, eliminating half of the elements, and comparing the target value to the value of the middle element of the remaining elements - until the target value is either found (and its associated element position is returned), or until the entire array has been searched (and "not found" is returned).

· Quick sort

Quick sort is a [comparison sort](https://en.wikipedia.org/wiki/Comparison_sort), meaning that it can sort items of any type for which a less-than relation (formally, a [total order](https://en.wikipedia.org/wiki/Total_order)) is defined. In efficient implementations it is not a [stable sort](https://en.wikipedia.org/wiki/Stable_sort), meaning that the relative order of equal sort items is not preserved. Quick sort can operate [in-place](https://en.wikipedia.org/wiki/In-place_algorithm) on an array, requiring small additional amounts of [memory](https://en.wikipedia.org/wiki/Main_memory) to perform the sorting.[Mathematical analysis](https://en.wikipedia.org/wiki/Analysis_of_algorithms) of quick sort shows that, [on average](https://en.wikipedia.org/wiki/Best,_worst_and_average_case), the algorithm takes [O](https://en.wikipedia.org/wiki/Big_O_notation)(*n* log *n*) comparisons to sort *n* items. In the [worst case](https://en.wikipedia.org/wiki/Best,_worst_and_average_case), it makes O(*n*2) comparisons, though this behaviour is rare.

Algorithm

Quick sort is a [divide and conquer algorithm](https://en.wikipedia.org/wiki/Divide_and_conquer_algorithm). Quick sort first divides a large array into two smaller sub-arrays: the low elements and the high elements. Quick sort can then recursively sort the sub-arrays.

The steps are:

· Pick an element, called a *pivot*, from the array.

· *Partitioning*: reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the *partition* operation.

· [Recursively](https://en.wikipedia.org/wiki/Recursion_(computer_science)) apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

The base case of the recursion is arrays of size zero or one, which never need to be sorted.

The pivot selection and partitioning steps can be done in several different ways; the choice of specific implementation schemes greatly affects the algorithm's performance.

**Modelling:**

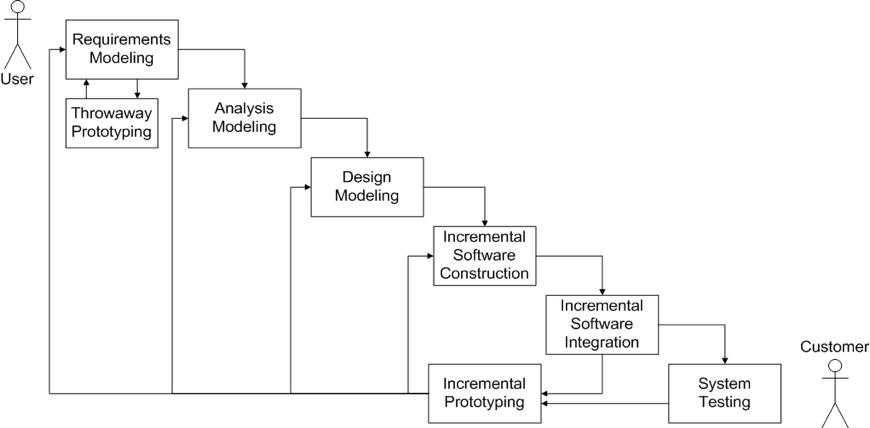
**1. COMET:**

· It stands for Concurrent Object Modeling and architectural design mEThod.

· COMET is a design method for UML supporting OO systems which is concurrent, distributed and real-time.

· It is compatible with USDP (Unified Software Development Process).

· Different phases of COMET are



· Requirements Modeling

· Use cases are generated, and serve as the requirements for the system.

· For binary search, sorted array is required.

· Also a key is to be given as input to be searched.

Add Functional Requirements

· Analysis Modeling

· Static Models

· Class Diagrams show the classes of the problem domain.

· Dynamic Models

· Show the problem domain objects participating in use cases.

· Design Modeling

· Software architecture is designed

· Object oriented architecture is being developed here.

· Problem Domain (Analysis Mode) is mapped to Solution Domain (Design Model)

· Subsystems are identified and structured

· Incremental Prototyping

· After modeling, the software subsystems are incrementally constructed and integrated into incremental prototypes.

· A phased, iterative approach

· Build and test a little at a time.

· If significant problems are testing, a return to the modeling phases is in order

· Incremental Software Construction

· It consists of

· Detailed design

· Coding

· Unit testing

· Incremental Software Integration

· During incremental software testing, the integration testing of each software is performed.

· Integration test for the increment

· based on the use cases selected

· developed for each use case

· Is type of white box testing (between object interfaces)

· Yields incremental prototype, formed by a software increment

· System Testing

· When the incremental prototype is to be turned over to the customer, it must first undergo system testing.

· The use cases of the requirements model are tested black box, ensuring the software meets specifications

· **Architectural Design**:

**Object Oriented Analysis and Design:**

**Object-oriented analysis and design** (**OOAD**) is a popular technical approach for analyzing, designing an application, system, or business by applying the object-oriented paradigm and visual modeling throughout the development life cycles to foster better stakeholder communication and product quality. According to the popular guide Unified Process, OOAD in modern software engineering is best conducted in an iterative and incremental way. Iteration by iteration, the outputs of OOAD activities, analysis models for OOA and design models for OOD respectively, will be refined and evolve continuously driven by key factors like risks and business value.Object-oriented modelling (OOM) is a common approach to modelling applications, systems, and business domains by using the object-oriented paradigm throughout the entire development. OOM is a main technique heavily used by both OOA and OOD activities in modern software engineering.

Object-oriented modelling typically divides into two aspects of work: the modelling of dynamic behaviours like business processes and use cases, and the modelling of static structures like classes and components. **OOA** and **OOD** are the two distinct abstract levels (i.e. the analysis level and the design level) during OOM.

**OOA:**

The primary tasks in object-oriented analysis (OOA) are:

· Find the objects

· Organize the objects

· Describe how the objects interact

· Define the behavior of the objects

· Define the internals of the objects

**OOD:**

During object-oriented design (OOD), a developer applies implementation constraints to the conceptual model produced in object-oriented analysis. Such constraints could include the hardware and [software](https://en.wikipedia.org/wiki/Software) platforms, the performance requirements, persistent storage and transaction, usability of the system, and limitations imposed by budgets and time. Concepts in the analysis model which is technology independent, are mapped onto implementing classes and interfaces resulting in a model of the solution domain, i.e., a detailed description of *how* the system is to be built on concrete technologies.

**Mathematical Model:**

Let S be the system such that :

S={s,e,X,Y,F,Sc,Fc}

Where,

s= initial state

e= end state

X= set of inputs

Y= set of outputs

F= set of function

Sc= Success cases

Fc= Failure cases

Let S’ be system in observation

Where S’ C S

S’ = {s,e,X,Y,F,Sc,Fc}

· S= start state

{init\_arr }

· e= end state

exit(0) ….success

· X= {(Xi,n ) | XiЄi, 0 ≤i< n }

· Y= {Y1, Y2, Y3} Є Y

Where ,

{Y1,Y2,}Є success

{ Y3} Є failure

· F= {F1, F2}

F1 = quicksort (x,st,end)

F2 = search (x,key,end)

· Sc= {Y1,Y2}

where Y1 = {key Є Xi , ji| 0 <= i<n ,i=j}

Y2={key Є Xi }

· Fc = {Y3}

Y3 = {keyЄXi ,ji | 0 <= i <n ,i != j}

**Design :**

**USE CASE:**

· It shows what a system will do and specifies external entities and various relationships.

· It has 3 components : Actors,Use Cases and Communication

· A use case defines the interactions between external actors and the system under consideration to accomplish a goal. Actors must be able to make decisions, but need not be human.

There are two types of actors : Human and System.

· There are 3 types of relationships between entities and UCD:

i. Association : between actors and use cases

ii. Dependency : between use cases and use cases

iii. Generalization : between use cases and use cases or actors and actors.

· Use cases can be of 2 types : Primary and Secondary.

**Usecase:**

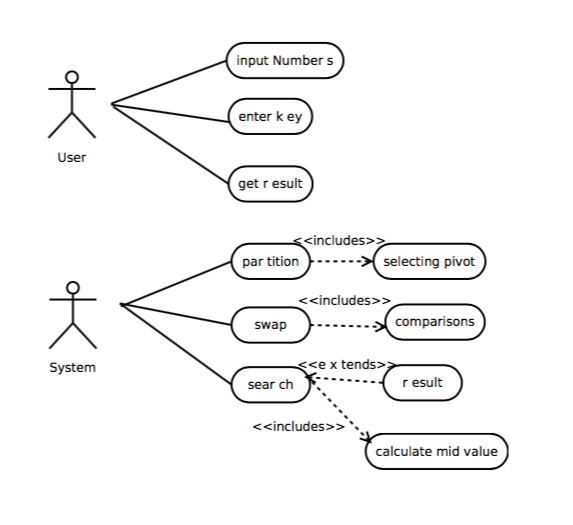


Fig: Use Case Diagram

**Algorithm:**

Algorithm BinarySearch(A,Key)

//A is array of size n

//Key is element to be searched

low = 1

high = n

while low<high do

mid=(low+high)/2

if A[mid] == key

then return mid

else if A[mid] < key

then low=mid+1

else

low=mid-1

end

end

**Input**. No. of elements , integers to be placed in array,key to be searched

**Output**: .position of key in array,if it is found.

**Platform**: Ubuntu 14.04

**Language:**Javascript

**Conclusion**: Hence studied and implemented software function for binary search for an unordered data using divide and conquer strategy and software modelling techniques.