International Journal of Latest Trends in Engineering and Technology
Vol.(8)Issue(4), pp.036-044
DOI: http://dx.doi.org/10.21172/1.84.05
e-ISSN:2278-621X

AN EMBEDDED AI SYSTEM FOR ENHANCEMENT OF COMPUTER LITERACY RATE WITH CASE STUDY IN INDIA

Somsubhra Gupta[1] and Ritadrik Chowdhury[2]

Abstract- Literacy means the ability to read and write. By that mean, a one - fourth of India's population is illiterate, according to the 2011 census. If the functional literacy is taken into consideration, one that extends beyond the ability of rudimentary reading and affixing a signature to paper, the experience says close to 40% of Indian people are functionally illiterate. On the other hand, due to the revolution in communications media in the past two decades, the entire world is gradually migrating to digital communication through mobile phones, computers, emails, internet, radio, television, tablets, touch screens, text messages, voice calls, and so on. This will lead to a better percentage of digitally literate people than those traditionally literate, especially if the use of cell phones is taken into consideration as a benchmark. In the recent past, it has been observed that less than 7% people live in India are belonging to the digitally illiterate class.

"Let's look at some of the prevailing scenario if 100% digital literacy is planned to be achieved before the next census takes place in 2021. According to some estimates, computer literacy in India is just 6.5%. The Internet penetration is still less than 10%. While radio and television reach is less than 150 million households, the mobile phone outreach is more than 53% of about 330 million households. It seems to be a hugely challenging task to target 330 million without institutionalization of the effort with allocated budget, however, a huge digital literacy penetration can be achieved if the sectors are targeted those live in the shadow areas of digital revolution. Panchayats (local governing bodies), for instance, have three million elected members and almost all of them are digitally illiterate. Involving the panchayati raj ministry, they can be targeted through 250,000 gramsabhas (village councils)"

Everybody needs an Operating System which is fast, reliable and user friendly. Ordinarily, the system is commonly used for few specific, trivial and predefined tasks aided by internet. Sometime the plug-ins is used to satisfy these job needs but not all the time or all the needs. E.g. the tasks need an easy access to any text editor or chart maker. These may be annexed with the need of entertainment viz. music or small ranged games. The browsers have close watch on several plug-ins which can be accessed by Computer literate classes only. The problem is, those operating systems are failed to take attentions on the field which are used for entertainment, relaxation and quick access for all. The presented work proposes an integrated intelligent System open to meet these criterions. The expected outcome of this work is a portable and handy Intelligent System, termed SURFER, that can provide the common users to execute common activities with decision making ability.

Keywords – Intelligent, Literacy, Pervasive, Prototype, Surfer, Ubiquitous.

I. INTRODUCTION

Technically, operating systems have been tightly related to the computer architecture, it is good idea to study the history of operating systems from the architecture of the computers on which they run. Operating systems have evolved through a number of distinct phases or generations which corresponds roughly to the decades.

[1] *Department of Information Technology JIS College of Engineering, Kalyani, W.B., India*
[2] *Deserve Sky Technology Kalyani, W.B., India*

The 1940's - First Generations

The earliest electronic digital computers had no operating systems. Machines of the time were so primitive that programs were often entered one bit at time on rows of mechanical switches (plug boards). Programming languages were unknown (not even assembly languages). Operating systems were unheard of.

The 1950's - Second Generation

By the early 1950's, the routine had improved somewhat with the introduction of punch cards. The General Motors Research Laboratories implemented the first operating systems in early 1950's for their IBM 701. The system of the 50's generally ran one job at a time. These were called single-stream batch processing systems because programs and data were submitted in groups or batches.

The 1960's - Third Generation

The systems of the 1960's were also batch processing systems, but they were able to take better advantage of the computer's resources by running several jobs at once. So operating systems designers developed the concept of multiprogramming in which several jobs are in main memory at once; a processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use.

For example, on the system with no multiprogramming, when the current job paused to wait for other I/O operation to complete, the CPU simply sat idle until the I/O finished. The solution for this problem that evolved was to partition memory into several pieces, with a different job in each partition. While one job was waiting for I/O to complete, another job could be using the CPU.

Another major feature in third-generation operating system was the technique called spooling (simultaneous peripheral operations on line). In spooling, a high-speed device like a disk interposed between a running program and a low-speed device involved with the program in input/output. Instead of writing directly to a printer, for example, outputs are written to the disk. Programs can run to completion faster, and other programs can be initiated sooner when the printer becomes available, the outputs may be printed.

Note that spooling technique is much like thread being spun to a spool so that it may be later be unwound as needed.

Another feature present in this generation was time-sharing technique, a variant of multiprogramming technique, in which each user has an on-line (i.e., directly connected) terminal. Because the user is present and interacting with the computer, the computer system must respond quickly to user requests, otherwise user productivity could suffer. Timesharing systems were developed to multiprogramming large number of simultaneous interactive users.

Fourth Generation

With the development of LSI (Large Scale Integration) circuits, chips, operating system entered in the system entered in the personal computer and the workstation age. Microprocessor technology evolved to the point that it becomes possible to build desktop computers as powerful as the mainframes of the 1970s. Two operating systems have dominated the personal computer scene: MS-DOS, written by Microsoft, Inc. for the IBM PC and other machines using the Intel 8088 CPU and its successors, and UNIX, which is dominant on the large personal computers using the Motorola 6899 CPU family.

A. Basic Concept

An **Operating system** is a intermediary agent between the user and the computer hardware.

- Manages the computer's resources (hardware, abstract resources, software)
- It's a resource allocator.
- It is also used to control programs to prevent errors and improper computer use.
- It is interrupt driven.

B. Operating System Benefits

- Simplifies hardware control for applications
- Enforcer of sharing, fairness and security with the goal of better overall performance
    - Trade-off between fairness and performance
    - Trade-off between optimal algorithms and lean algorithms in OS is overhead.

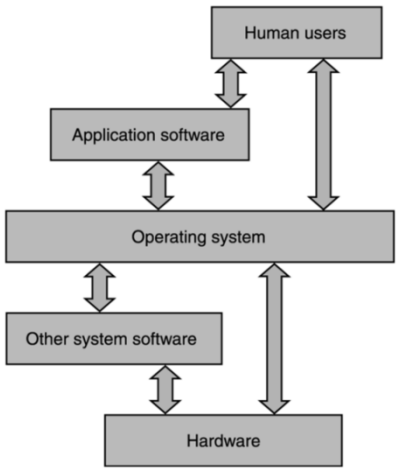


Figure 1: Users / Processes access the Computer's resources through OS

Provides abstract resources

- Sockets
- Inter-process communication

C. Overview of SURFER

This Intelligent System has been introduced keeping in the mind the need of most commonly used features with an exception of decision making by the device itself. This is introduced in a portable and transferable way sans all features of a Desktop /Laptop computer. As such, it is not exactly be identified as a Tab/ Mobile phone OS since they incorporate versatile scope and hence complex. Rather, this may be resembled with a Modular Object Oriented Dynamic Learning Environment (MOODLE) in terms of the task that can be carried out through it.

What's New in Surfer?

After a short survey from the users, the **Surfer** has been develped. An ultimate resource from where a person can fulfil his/her needs. Surfer is becoming one of his/her best friends. It not only takes care of browsing part but also helps to create chart, relaxation, editing securely and fast.

- **Chart Creator:**
  Surfer contains a button which redirects the user to a certain chart maker window where a user can put the values and creates a chart as per his need (Refer to Figure 1.1.).
- Example: If someone wants a graphical representation of a marks-assessment, he/she can easily put the names and marks on a given field and can see a created chart as per his/her need.
- **Inbuilt Text Editor**
  Surfer contains a button which redirects the user to a certain text editor window where a user can write and save texts as per his need.

Figure 2: Chart

- **Inbuilt Paint Editor**
  Surfer contains a button which redirects the user to a certain paint window where a user can draw and save image files as per his need.
- **Inbuilt Browser**
  Surfer contains a button which redirects the user to a certain converter window where a user can browse the internet.
- **Inbuilt Entertainer**
  Surfer contains a button which redirects the user to a certain window where a user can play audio or can play some online small games for his/her relaxation.*
- **Inbuilt Protector**
  Surfer contains a button which redirects the user to a certain window where a user can scan his/her local discs to find any attacks, malware.
- **Inbuilt Tutor**
  Surfer contains a button which redirects the user to a certain window where a user can have the help of an Artificial Intelligence.

II. TECHNOLOGY AND PROGRAMMING

D. Why C# and Visual Studio with COSMOS

Cosmos (C# Open Source Managed Operating System) is an operating system development kit which uses Visual Studio as its development environment. Despite C# in the name, any .NET based language can be used including VB.NET, FORTRAN, Delphi Prism, Iron Python, F# and more. Cosmos itself and the kernel routines are primarily written in C#, and thus the Cosmos name. Besides that, NOSMOS (.NET Open Source Managed Operating System) sounds stupid.

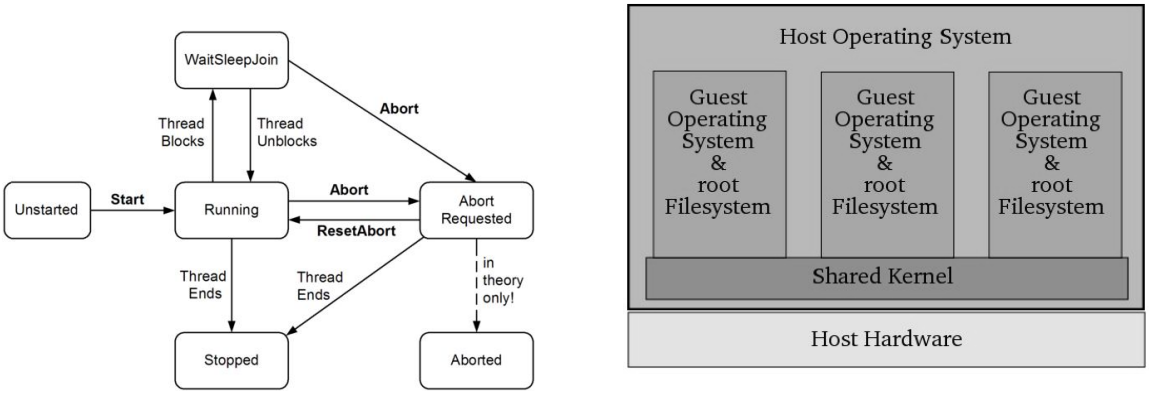


Figure 3: Architecture and Control Flow

E. Choice of Model

The prototype model has been preferred over other model because of

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified
- Requirements validation, Quick implementation of, incomplete, but functional, application.

F. Console Programming

Consoles manage input and output (I/O) for character-mode applications (Applications that do not provide their own graphical user interface). Under Windows, the console is always a window that resembles the Command Prompt window. You can open and read and write from/to that thing in your windows program. It's not a buffer or a text file, but you can write a buffer or text file and then transfer that entity to the console

G. Basic cosing and C# code snapshot

To read a line of text from the console window, you use the **Console.ReadLine()** method. This will read an input stream from the console window and return the input string when user presses the Enter Key.

```
static void Main()
{
    // Boot the Cosmos kernel:
    Cosmos.Sys.Boot xBoot = new Cosmos.Sys.Boot();
    xBoot.Execute();

    Console.WriteLine("Cosmos booted successfully");
    Console.WriteLine("One");
    Console.WriteLine("Two");
    Console.WriteLine("Three");
    string xResult = Console.ReadLine();
    Console.Write("Text typed: ");
    Console.WriteLine(xResult);
}
```

There are also two methods for writing to the console, which are used extensively

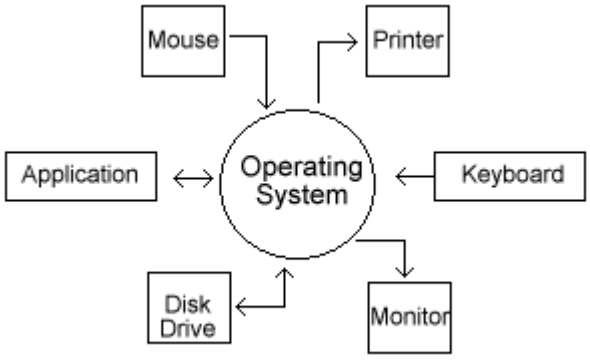
- **Console.Write()** — Writes the specified value to the console window.
- **Console.WriteLine()** — This does the same, but adds a newline character at the end of the output.

H. Requirenment Specification

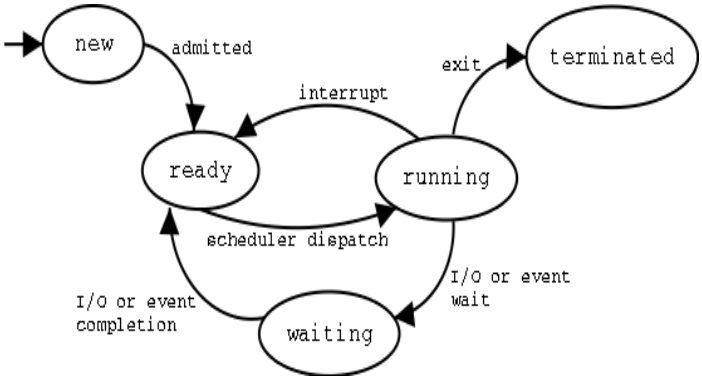
Preferred Framework:	ASP. Net
Preferred Language:	C#
Preferred Data Base:	SQL Server 2008
Preferred Operating System:	Microsoft Windows 7
Preferred Console Application:	COSMOS
Physical Memory:	2048 MB
Hard Disc Capacity:	20GB
Processing Speed:	3.2 GHz

I. Data Flow Diagram

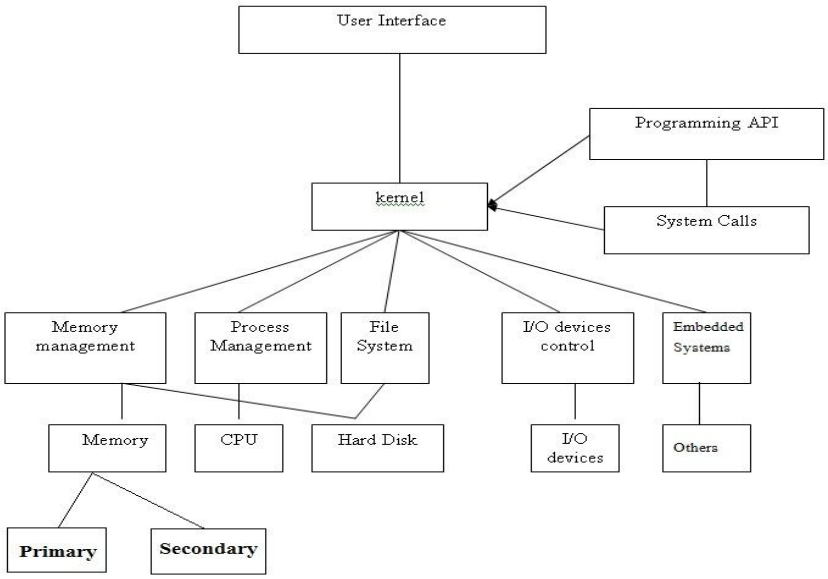
The Flow diagrams are presented in the following - Level wise



(a)



(b)



(c)

Figure 3: Data Flow Diagrams (a) level 0 (b) level-1 and (c) level -2

IV. Embedment of Intelligence and Decision Making

Artificial intelligence is the branch of computer science concerned with making computers behave like humans. The term was coined in 1956 by John McCarthy at the Massachusetts Institute of Technology. The application of Intelligent Systems is expressed in brief in the following:

A. Intelligent System Applications

Available are the machines that can play master level chess for a few hundred dollars. There is some AI in them, but they play well against people mainly through brute force computation--looking at hundreds of thousands of positions. To beat a world champion by brute force and known reliable heuristics requires being able to look at 200 million positions per second. Here are some mentionable areas with breakthrough in intelligent computing:

Speech recognition

In the 1990s, computer speech recognition reached a practical level for limited purposes. Thus United Airlines has replaced its keyboard tree for flight information by a system using speech recognition of flight numbers and city names. It is quite convenient. On the other hand, while it is possible to instruct some computers using speech, most users have gone back to the keyboard and the mouse as still more convenient.

Understanding natural language

Just getting a sequence of words into a computer is not enough. Parsing sentences is not enough either. The computer has to be provided with an understanding of the domain the text is about, and this is presently possible only for very limited domains.

Computer vision

The world is composed of three-dimensional objects, but the inputs to the human eye and computers' TV cameras are two dimensional. Some useful programs can work solely in two dimensions, but full computer vision requires partial three-dimensional information that is not just a set of two-dimensional views. At present there are only limited ways of representing three-dimensional information directly, and they are not as good as what humans evidently use.

Expert systems

A ``knowledge engineer'' interviews experts in a certain domain and tries to embody their knowledge in a computer program for carrying out some task. How well this works depends on whether the intellectual mechanisms required for the task are within the present state of AI. When this turned out not to be so, there were many disappointing results. One of the first expert systems was MYCIN in 1974, which diagnosed bacterial infections of the blood and suggested treatments. It did better than medical students or practicing doctors, provided its limitations were observed. Namely, its ontology included bacteria, symptoms, and treatments and did not include patients, doctors, hospitals, death, recovery, and events occurring in time. Its interactions depended on a single patient being considered. Since the experts consulted by the knowledge engineers knew about patients, doctors, death, recovery, etc., it is clear that the knowledge engineers forced what the experts told them into a predetermined framework. In the present state of AI, this has to be true. The usefulness of current expert systems depends on their users having common sense.

Heuristic classification

One of the most feasible kinds of expert system given the present knowledge of AI is to put some information in one of a fixed set of categories using several sources of information. An example is advising whether to accept a proposed credit card purchase. Information is available about the owner of the credit card, his record of payment and also about the item he is buying and about the establishment from which he is buying it (e.g., about whether there have been previous credit card frauds at this establishment).

In the proposed work, in course of decision making, The Hill Climbing Algorithm has been used so as to confer decision that’s work in tandem with prototype model.

B. Introducing Hill Climbing

Hill climbing is an optimization technique for solving computationally hard problems. It is best used in problems with “The property that the state description itself contains all the information needed for a solution” (Russell & Norvig, 2003). The algorithm is memory efficient since it does not maintain a search tree: It looks only at the current state and immediate future states.

Hill climbing attempts to iteratively improve the current state by means of an evaluation function. In contrast with other iterative improvement algorithms, hill-climbing always attempts to make changes that improve the current state. In other words, hill-climbing can only advance if there is a higher point in the adjacent landscape.

This is a variety of depth-first (generate - and - test) search. A feedback is used here to decide on the direction of motion in the search space. In the depth-first search, the test function will merely accept or reject a solution. But in hill climbing the test function is provided with a heuristic function which provides an estimate of how close a given state is to goal state.

Algorithm

1. Evaluate the initial state. If it also a goal state return it and quit. Otherwise continue with the initial state as current state.
2. Loop until a solution is found or until there are no new operators left to be applied over current state.
    a) select an operator that has not been applied to the current state and apply it to produce a new state.
    b) Evaluate the new state-
        i) If it be a goal state then return it and quit.
        ii) If it not be a goal state but it is better than the current state, then make it the current state.
        iii) If it is not better than the current state then continue on the loop.

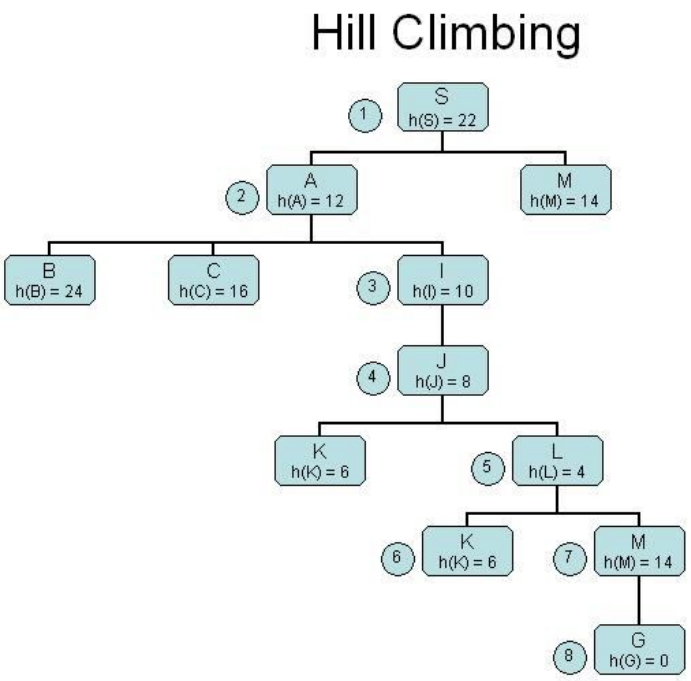


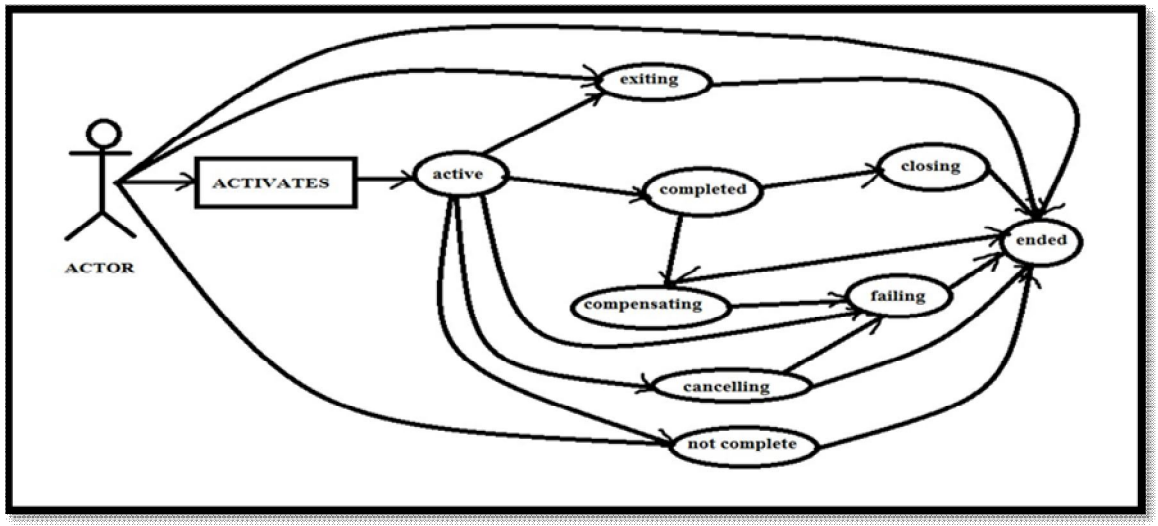
Figure 4: Hill Climbing graphical representation

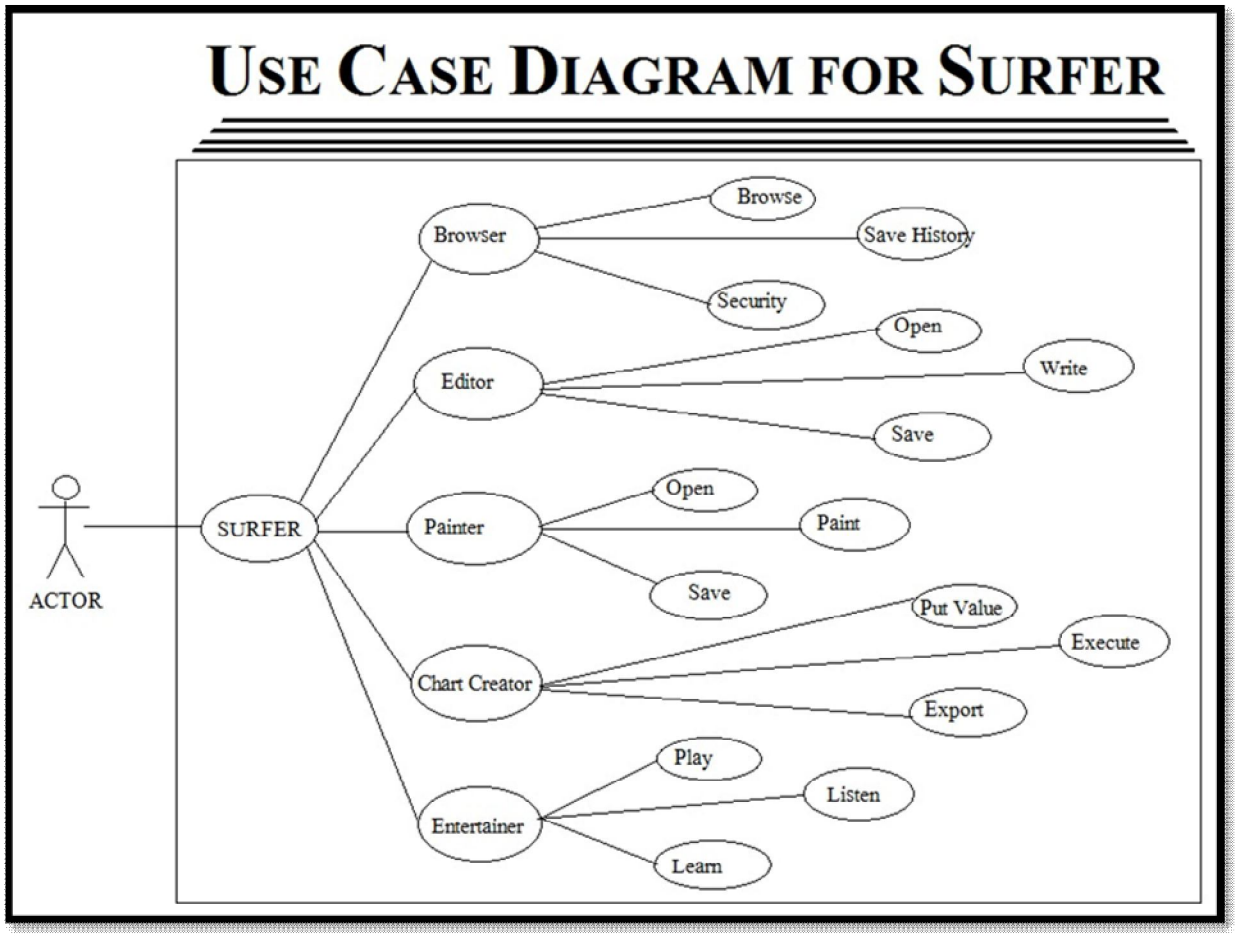
Pseudo Code

```
function HILL-CLIMBING(problem) returns a solution state
         inputs: problem, a problem
         static: current, a node
                 next, a node
         current <— MAKE-NODE(INITIAL-STATE[problem])
         loop do
                 next— a highest-valued successor of current
                 if VALUE[next] < VALUE[current] then return current
                 current *—next
         end
```

C. Sequence Diagram

The sequence diagram has been presented in the following





D. Complexity and Overheads

Since the evaluation used looks only at the current state, hill-climbing does not suffer from computational space issues. The source of its computational complexity arises from the time required to explore the problem space.
Random-restart hill-climbing can arrive at optimal solutions within polynomial time for most problem spaces. However, for some NP-complete problems, the numbers of local maxima can be the cause of exponential computational time. To address these problems some researchers have looked at using probability theory and local sampling to direct the restarting of hill-climbing algorithms. (Cohen, Greiner, & Schuurmans, 1994).

IV.CONCLUSION

The SURFER is a system which can help the beginners to explore the world with its browsing facility and explore his/her knowledgebase which will be trained by the A.I. Tutor. This system not only guide him/her to do the job and learn the specific areas but also it will show some video tutorial sometimes to solve the toughest moments which can be faced by any Computer Beginner. End of the day, the SURFER will be the friend, philosopher and guide to the beginner.

REFERENCES

[1] N. Banerjee, S. Rollins, and K. Moran. Automating Energy Management in Green Homes. In SIGCOMM Workshop on Home Networks (HomeNets), 2011.

[2] Angela Demake Brown, Todd C. Mowry and Orran Krieger, Compior Based I/O prefetching for out-of-core applications. ACM Transactions on Computer Systems, 19(2): 111-170, 2001.

[3] Modern Trends Used In Operating Systems For High Speed Computing Applications , (IJCSE) International Journal on Computer Science and Engineering, Vol. 02, No. 05, 2010, 1514-1523.

[4] Sergey Bykov, Alan Geller, Gabriel Kliot, James Larus, Ravi Pandya, and Jorgen Thelin "Orleans: Cloud Computing for Everyone", Proceedings of the 2nd ACM Symposium on Cloud Computing (2011).

[5] Polakovic, J., Ozcan, A. E., Stefani, J.-B.: Building reconfigurable component-based OS with THINK, Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, Dubrovnik, Croatia, pp. 178 – 185, 2006

[6] TinyOS community forum, http://www.tinyos.net/

[7] Caromel, D., Delb´e, C., di Costanzo, A., Leyton, M.: ProActive: an integrated platform for programming and running applications on grids and p2p systems, Computational Methods in Science and Technology, issue 12, pp. 69 – 77, 2006

[8] Bures, T., Hnetynka, P., Plasil, F.: SOFA 2.0: Balancing Advanced Features in a Hierarchical Component Model, Proceddings of SERA 2006, Seattle, USA, IEEE CS, 2006

[9] Sub-Operating Systems: A New Approach to Application Security by Sotiris Ioannidis, Steven M. Bellovin, Jonathan M. Smith published at https://www.cs.columbia.edu/~smb/papers/subos.pdf

[10] http://www.livemint.com/Industry/XmQNrsrg8zWfOFKHuMDtfJ/Recipe-for-100-digital-literacy-before-2021.html

[11] http://programcall.com/2/csnet/what-is-console-application.aspx