IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-5, May 2019 E-ISSN: 2347-2693

# Modelling of Knowledge Based Transport Vehicle System using Reliable Software System

**Mohd Ashraf[1*], Z. Hussain[2] V. Singh[3]**

[1] Department of Computer Science & Engg, Maulana Azad National Urdu University, Hyderabad, India
[2] Department of Information Technologies, Maulana Azad National Urdu University, Hyderabad, India
[3]Cloud Technologists, Nokia Networks, Gurgaon, India

*Corresponding Author: ashraf.saifee@gmail.com, Tel.: +91-9494147875*

DOI: https://doi.org/10.26438/ijcse/v7i5.251256 | Available online at: www.ijcseonline.org

Accepted: 12/May/2019, Published: 31/May/2019

***Abstract***— The technological knowledge and experiences are the key elements in designing the transportation vehicles, thereby enhancing it's safe operations. This research paper introduces a model which can be used for developing a safe and reliable transportation system with smart vehicle design. Requirements in terms of both software implementations and vehicle design have been proposed in this research paper. Knowledge Base technique is the foundation of technical design reuse. Implementation of this technique can be relied on the software. Two distinct algorithms are used in this paper to improve upon the safety, reliability and quality of a system. The difference between the results of the algorithms would help us in concluding the best technique.

***Keywords***— Knowledge Engineering, Reliable Software, Ontology, Sensor System, Computer-aided-technology

## I. Introduction

Transportation system is the most useful and integral part of human life. Everyday technological advances are helping us achieve a better transportation system. However, there are still many areas where more interference is required both on technology and experiential learning's. The research on "Knowledge-Based Transport Vehicle System using Reliable Software System" is another leap in the transportation world, making your every journey safe, reliable and of the finest quality. Transport vehicle system has many sub-systems like as fire control system, protection system, power system, fuel transmission system, electronic system, sensor system, and more features are included in it. So that transport vehicle design can be included in many fields and in multi-faceted technical knowledge and experience. This technical knowledge and experience can be effectively included as the key to increase transport development vehicle capability, maintainability and quality, It is decreased the time and economy uses of the development cycle. In the recent scenario, systems provide the geometric modeling functionality which facilitates the drafting operations of transport vehicle design but do not provide designers with the necessary knowledge to develop good transport vehicle designs. Therefore, conventional computer-aided design technology is unsuitable for processing empirical type of knowledge which is critical in the transport vehicle design problems. The importance of knowledge engineering in engineering successful designs has been recognized by the researchers lately.

Many knowledge management methodologies have been presented for different design areas. The drafting operations of transport vehicle design but does not provide designers with the necessary knowledge to develop good transport vehicle designs. However, there is no effective and feasible knowledge management tool to aid the development of transport vehicle development in the current scenario. In this paper, a methodology of construction of knowledge-based engineering platform for a transport vehicle is proposed to support transport vehicle design. The main design process for Transport vehicle and its architecture is designed using the software of the system. The design of the software system is more reliable for the existing used systems.

So focusing on the construction of various types of the knowledge base, a case study is presented in this paper which describes the knowledge base engineering platform for the transport vehicle and reliable software system. Further, the design of the armored vehicle is produced based on the various methodology of knowledge engineering architecture of the knowledge-based engineering platform and category of armored vehicle knowledge then the modeling, storage, and retrieval for each type of knowledge are illustrated. The ontology is adopted to create the innovative design knowledge base combing the case-based reasoning (CBR) with function-structure mapping method to construct the reuse design knowledge base. Design process integration technique is applied to build the typical design problem-solving knowledge base. KADS's [5]. Guoxin Wang et. al discussed the implementation of distinct software in the vehicle control unit of an automated Vehicle. Each module in the vehicle control unit which is considered to be safety critical is performed by two sets of distinct algorithms in pseudo-parallel (one after the other) in the same microprocessor. The output of the distinct algorithms detects common-mode software errors and common mode hardware failures in the two processors, thus improving the safety of the vehicle control unit.

This paper deals with various aspects of the modeling of knowledge base transport vehicle system. In Section 2, we discussed the proposed model. Section 3 describes the algorithm, Section 4 deals the implementation of system, and section 5 explores conclusion and future scope.

## II. Proposed Model

The proposed model is a combination "Hardware Module" and "Software Module". The primary elements of Hardware module are Glass, Door, Electrical fixatures and Lightings, Sheeting, Climate control, Wheel, Standard features, Interior features and exterior features. While, the Software Module is a combination of Analysis, Requirements, Codes, Implementation, Testing, Reuse, Validation and Maintenance. The synergy of the two modules is the key to a reliable and safe transportation system and of course is foundation of knowledge-based modeling.

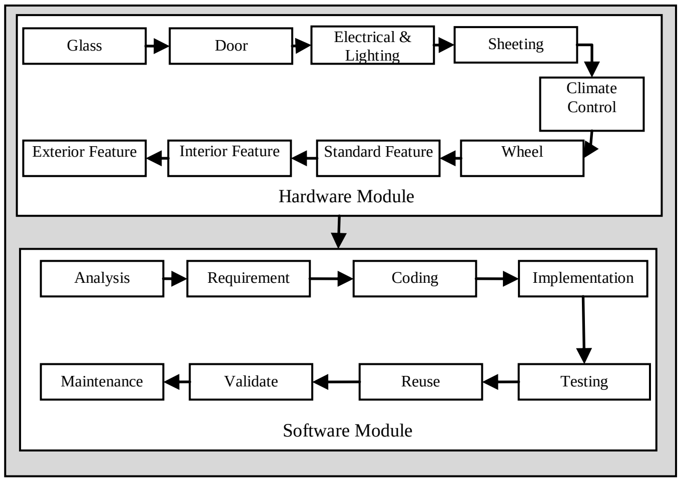

**Figure 1:** Model of Knowledge based Transport Vehicle System

## III. Proposed Algorithm

The algorithm for knowledge-based transport vehicle is divided into two modules and is represented in pseudo code. The output of Module in one of the algorithm is a set of the bug reports. Each bug report identifies the information about the failure, the set of input under which failure and module take the input as a constraint from module one output short the input constraint that expose in module one.

### *Module-1*

Program p, knowledge data k, output oracle O
**Result:** failure report f
F-set of (failure, set of variable, sets of input)

The inputs to the algorithm are: a program P and an output oracle O. The output of the algorithm is a set of bug reports B for the program P, according to O. Each bug report contains: identifying information about the failure, the set of all inputs under which the failure was exposed, and the set of all path constraints that lead to the inputs exposing the failure

```
Algorithm:
    {
        p=set of domain knowledge data;
        f=0;
        Pc Queue=empty Queue ( );
    }
Enqueue (pcqueue, emptypathvariable());
```

The algorithm uses a queue of path constraints. A *path constraint* is a conjunction of conditions on the program's input parameters. The queue is initialized with the empty path constraint

```
    {
    While not empty (pcQueue) and not time Expired()
    }
    Do
        {
        Path Variable=dequeue (Pc Queue);
```

The algorithm uses a constraint solver to find a concrete input that satisfies a path constraint taken from the queue

```
        }
        Input=solve (Path Variable);
        {
        if input + k Then
        }
```

The program is executed concretely on the input and tested for failures.

```
Output=execute (p.input);
Failures=get Failures (f.output);
```

The path constraint and input for each detected failure are merged into the corresponding bug report

```
    {
        For each f in failures
```

Next, the program is executed symbolically on the same input

```
        do
        merge {f, path Variable, input} into f;
```

The result of symbolic execution is a path constraint, V*ni*=1 *ci*, that is fulfilled if the given path is executed (here, the path constraint reflects the path that was just executed). The algorithm then creates new test inputs by solving modified versions of the path constraint

```
        }
                $c_1$^….$c_n$=execute (p,input);
```

For each prefix of the path constraint, the algorithm negates the last conjunct (line 15

```
        {
                for each i=1,…..,n
        }
        do
        newPC=$c_1$^……$c_{i-1}$^$c_i$;
        enqueue ($P_c$ Queue, newPC);
        Return f;
```

***Module-2***

Parameters: program p, oracle o,bug report b
Result: short path constraint that expose b.failure
For a given bug report *b*, the algorithm first intersects all the path constraints exposing *b.failure* (line1).

```
C1^……^$c_n$: =intersect (b.pathConstraints);
Pc: =true;
```

This eliminates irrelevant constraints, and a solution for a shorter path constraint is often a smaller input The minimizer systematically removes one condition at a time

```
{
        For each i=1…..n do
        $P_{ci}$:=$c_1$^…..$c_{i-1}$^$c_{i+1}$^…..$c_n$;
}
Input:=solve($pc_i$);
If input / then
Output:=execute Concrete (p.input);
Failures;=get Failure(o,output);
If b.failure/failures then
Pc:=pc^$c_i$
$Input_{pc}$:=solve(pc);
If $input_{pc}$/ then
$Output_{pc}$:=execute Concrete(p,$input_{pc}$);
If b.failure ϵ $failures_{pc}$ then
Return pc;
Return shortest (b.pathConstraint)
```

A solution, if it exists, to such an alternative path constraint corresponds to an input that will execute the program along with a prefix of the original execution path, and then take the opposite branch. The path constraint minimization algorithm is used here. This method *intersects,* returns conjunction containing the conditions that are present in all given path constraints. A While the method, *shortest,* returns the path constraint with fewest conjuncts. The other auxiliary functions are the same as in Figure 1. The failure detection algorithm returns bug reports for different failures. Each bug report contains a set of path constraints leading to inputs and exposing the failure. Previous dynamic test generation tools presented the whole input to the user without an indication of the subset of the input responsible for the failure. As a postmortem phase, our minimizing algorithm attempts to find a shorter path constraint for a given bug report. If one of these shorter path constraints does not expose *b.failure*, then the removed condition is required for exposing *b.failure*. The final path constraint is the conjunction of all such required conditions. From the minimized path constraint, the algorithm produces a concrete input that exposes the failure.

The algorithm does not guarantee that the returned path constraint is the shortest possible that exposes the failure. However, the algorithm is simple, fast, and effective in practice. Each failure might be encountered along with several execution paths that might partially overlap. Without any information about the properties of the inputs, delta debugging minimizes only a single input at a time, while our algorithm handles multiple path constraints that lead to a failure

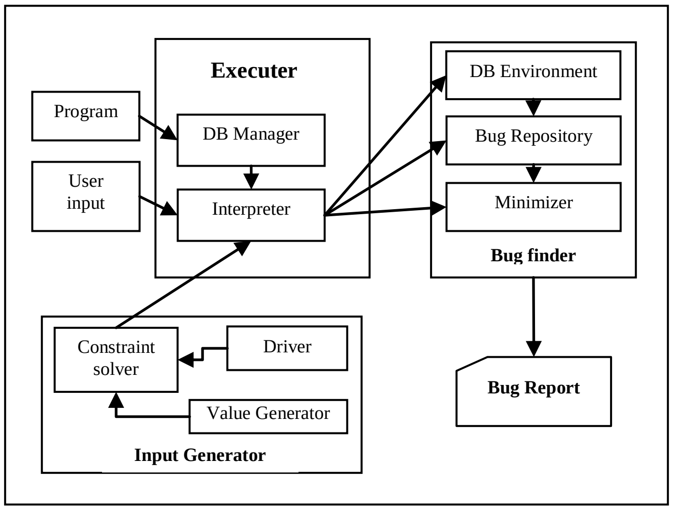

**Figure 2**: Diagram for Bug finding process

## IV. IMPLEMENTATION

The application methods that are used as software in vehicle design is to identify which elements of the specification were used in safety gaps. All software users require safety gap elements in the software. The ordinary generation of motion control commands is not considered to be safety measurements. However, the need for those motion control commands is for emergency, safety and critical situation. These motion control rules are used in immediate response emergency stopping system. The motion control laws are dependent upon speed and position of the vehicle. The speed and position of the vehicle are counted by odometer pulse from the wheels. The counting is done in a preprocessor for every wheel and every main processor.

The error detection is calculated by an extra pulse detection algorithm. The back-up of the error detection is done through the main processor with cross axle drift analysis for centerline speeds of the wheel pair combinations. A combination of the differential pulse count and the differential latched time count is multiplied by a calibration factor to get the measured centerline speed. The first algorithm for hardware measurement error detection and speed detection compares the speed measured for every cross axle wheel pairs, through the wheel by wheel check for differential pulse count. The rate of change and ranges are checked for the calibration factors and a check of the condition of the circumstances under the calibration factor is also calculated.

These Fault detection algorithms are used to satisfy the requirement of proposed software redundant which can be constantly improved.

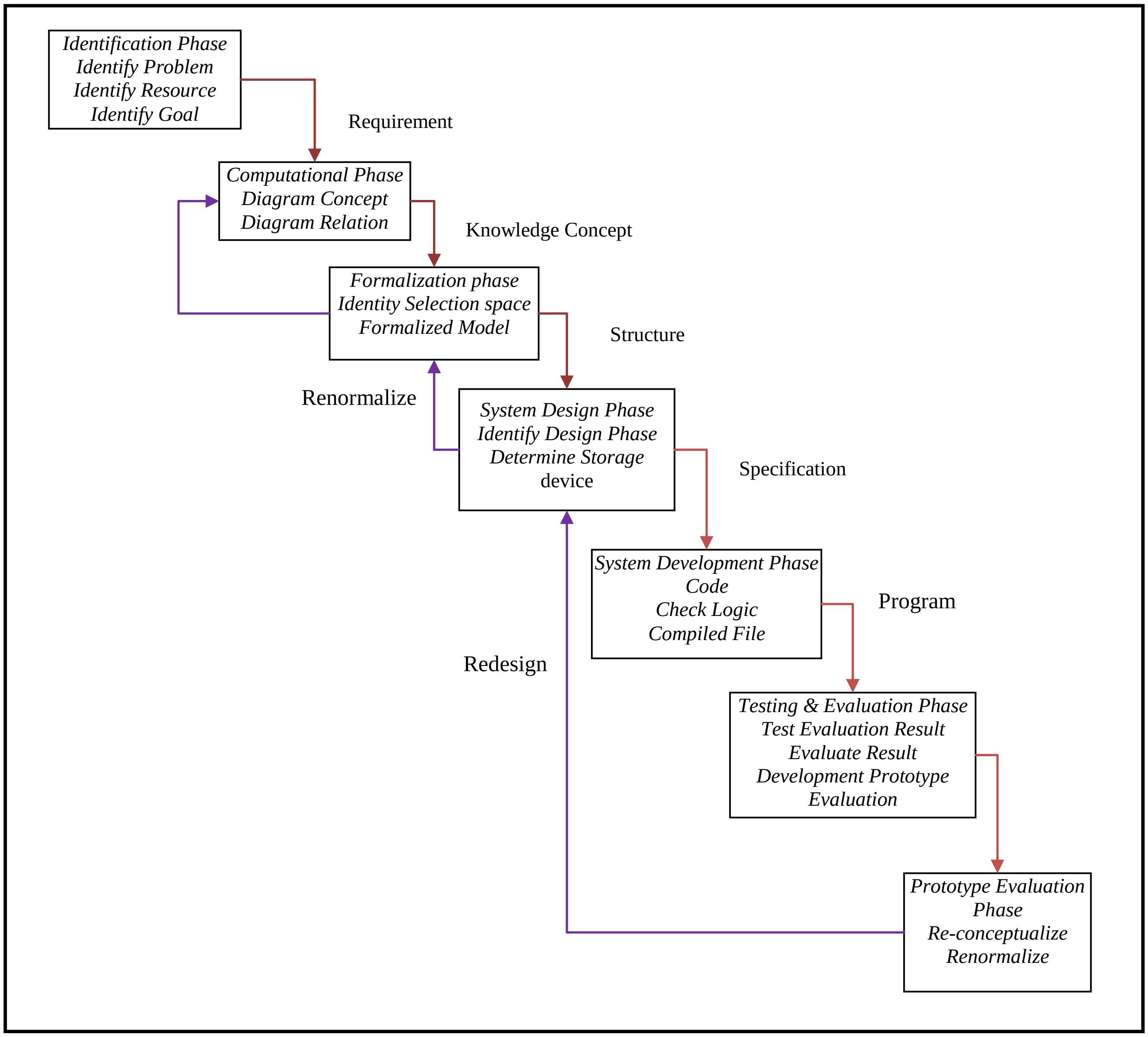

**Figure 3:** Process of Implementation of transport Vehicle system

However, a better approach is to compare the total measure of the speed with the original output which results in the total measure collected by the differential pulse counts, a fixed time differential and an ideal wheel Alignment factor. An average of the differential pulse counts from all four wheels which are replaced one by one and the two measures of cross axle pairs. The disparity detector must allow for the uncertainty or jitter introduced so that the sanity of the primary speed measure is verified. This method provides a simple confidence check of the speed measurement and an indirect confidence check of the calibration factor used for position calculation. The other major element used by the control laws is the calculated commanded speed. The differences between the commanded, and measured speed is that the position dictates the torque commands to the propulsion and brake systems. The complexity and fine-tuning are built into commanded speed algorithms which would limit a backup algorithm to just change the order of computation and rescaling. To avoid this limitation, the scope of the backup has been restricted to that part of the commanded speed logic which is used only in an emergency stop. A worst case study of the profile of speed under emergency conditions has shown that safety is assured if the measured deceleration rate of the vehicle remains within the range of a fixed emergency deceleration rate.

The deceleration rate which is expected under emergency conditions is not fixed, but the jerk is limited for safety reasons. To make the backup check work, the measured and fixed deceleration comparison is held off until; the time for worst case jerk profiling has passed. The differential of the measured speed would do a direct comparison between differential and desired deceleration rate, which would force the tolerance to be too large to assure a safe stopping distance.

The solution is to use a running average of the differential after the jerk time elapsed and use a diminishing tolerance in the comparison. The jitter in the measured deceleration should sum to zero over time, so that the average measured deceleration rate should quickly approach the expected value. And the diminishing tolerance approaches zero. If the measured deceleration breaches the ceiling of the expected value plus tolerance, then the open-loop braking system would be activated in time to assure the safe stopping distance. Sometimes there is some defective information collected in the system in gathering of all the data.

## V. CONCLUSION AND FUTURE SCOPE

To support the transport vehicle design and reliability of the software, a transport vehicle of knowledge-based engineering is constructed which is the foundation of this research. Three unique steps have been formulated i.e., innovation design knowledge, reuse design knowledge, solving process knowledge and failures in software that is proposed. This is how we have classified the design knowledge of transport vehicle. The most robust system of the entire design is the software since it provides the maximum protection in an operable system. The features are designed and divided to first detect anomalies in the hardware, second, to protect against common mode hardware failures and software errors. While the third feature assures that failure will be detected by the first two features before a second failure has time to occur. Unity is the most powerful equation to solve reliability issues. Reliability is the probability that an item will perform a required function under stated conditions for a stated period of time. Unity is the addition of Probability of Survival and Probability of Failure. Satisfactory and Unsatisfactory operation (failure) both frame the required function in the stated period of time.

## REFERENCES

[1] J. Vijaya Sagar Reddy & G. Ramesh "*Failure Detection Algorithm for Testing Dynamic Web Applications Department of CSE*", International Journal of Computer & Communication Technology, Volume-**5**,Issue-**4**,pp **42-46, 2016**.

[2] Gilb, Tom, *"Distinct Software*" ACM SIGSOFT, Software Engineering Notes, Vol. **6**, No. **2**, p. **17 1981**

[3] Fischler, M. A., 0. Firschein, D. L. *Drew "Distinct Software: An Approach to Reliable Computing"* in proceeding Second USA-JAPAN Comnference, pp **573-579, 1975**

[4] R. Solanki, "*Principle of Data Mining*", McGraw-Hill Publication, **India**, pp. 386-398, 1998.

[5] Wang, Guoxin & Yan, Yan & Hu, Lichen & Zhang, Xiang & Wang, Lu. (2009). *Construction of knowledge based engineering platform for armored vehicle.* IEEM 2009 - IEEE International Conference on Industrial Engineering and Engineering Management. 10.1109/IEEM.2009.5372938. IEEM **2009**

[6] Y. J. Chen, Y. M. Chen, H. C. Chu and H. Y. Kao, *"On technology for functional requirement-based reference design retrieval in engineering knowledge management,"* Decision Support Systems, vol**. 44,** no. **4,** pp. **798-816, 2007**.

[7] Y. J. Chen, Y. M. Chen and H. C. Chu, "Enabling collaborative product design through distributed engineering knowledge management," Computer in Industry, vol. **59**, no.**4**, pp. **395-409, 2008**.

[8] C. K. Mok, K. S. Chin and H. Lan, *"An Intirnate-based intelligent design system for injection moulds"* Robotics and Computer-Integrated Manufacturing, vol. **24**, no. **24**, pp**. 1-15, 2008.**

[9] L. F. Lai, "*A knowledge engineering approach to knowledge management,"* Information Sciences, vol. **177**. no.**177**, pp. **4072-4094, 2007.**

[10] S. C. Brandt, J. Morbach, M. Miatidis, M. Theiben, M. Jarke and W. Marquardt, *"An ontology-based approach to knowledge management in design processes,"* Computers and Chemical Engineering, vol. **32,** no. **32**, pp. **320-342, 2008**

[11] R. I. M. Young, A. G. Gunendran, A. F. Cutting-Decelle and M. Gruninger, *"Manufacturing Knowledge Sharing In Plm: A Progression Towards The Use Of Heavy Weight Ontologies"* International Journal of Production Research, vol. **45**, no. **7**, pp. **1505-1519, 2007**

[12] F. Gailly, W. Laurier, G. Poels, *"Positioning and formalizing the REA enterprise ontology"* Journal of information systems, vol. 22, no. **2**, pp. **219-248, 2008**.

[13] D. Baxter, J. Gao and K. Case. *"A framework to integrate design knowledge reuse and Requirements management in engineering design"* Robotics and Computer-Integrated Manufacturing, vol. **24**, no**. 4,** pp. **585-593, 2008**

[14] K. C. Ku, A. Wensley, H. P. Kao, "Ontology-based knowledge management for joint venture Projects" Expert Systems with Applications, vol. **35**, no. **1-2**, pp. **187-197, 2008.**

[15] K. Mohan, R. Jain, B. Ramesh, *"Knowledge networking to support medical new product development"* Decision Support Systems, vol. **43**, no. **4**, pp. **1255-1273, 2007**

## Authors Profile

*Dr. Mohd Ashraf* pursed Bachelor of Technology in Computer Engineering from Gobind Bhallabh Pant University of Agriculture & Technology, Pantnagar (UK) in year 2004 and Master of Technology in Computer Science & Engineering , Aligarh Muslim University, Aligarh India in year 2009. He completed his Ph.D in the field of Computer Science & Engineering from Gautam Budhha University in year 2014. and currently working as Associate Professor in Department of Computer Science & Engineering, Maulana Azad National Urdu University , Hyderabad since 2015. He has published more than 40 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. His main research work focuses on Optimization Algorithms, Network Reliability, Soft Computing, Fuzzy logic. He has 14 years of teaching experience and 5 years of Research Experience.

*Dr. Md. Zair Hussain* pursued bachlor and master degree in . He completed his Ph.D in the field of Information Technology from NIT Patna. He is currently working as a associate professer in the department of Information Technology, Maulana Azad National Urdu university Hyderabad.. He has published more than 15 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it's also available online. He has more than 20 years of teaching experience .

*Ms. Vrinda Singh* is a MTech in VLSI and B.Tech in Electronics and Communications. She completed her integrated course in the year 2015. Currently, she is working as Cloud Technologist in Nokia networks, Delhi. Her research area are cloud computing Network security, IOT etc. She is having more than 4 year research experiences.