Journal of Engineering Education Transformations,
Volume No. 37, January 2024 Special Issue, eISSN 2394-1707

Students Unlock the Power of Real Systems: An Experiential Learning in System Software Course

Nagaratna D Kulenavar[1], Sujatha C[2], Umadevi F M[3], Indira B[4], G S Hanchinamani[5], Jayalaxmi G.N[6]
[1,2,3,4,5,6]School of Computer Science & Engineering, KLE Technological University, Hubballi, Karnataka
[1]kulennavar@kletech.ac.in , [2]sujata_c@kletech.ac.in , [3]uma_devi_fm@kletech.ac.in
[4]indira_bidari@kletech.ac.in ,[5]gs_hanchinamani@kletech.ac.in , [6]jaya_gn@kletech.ac.in

***Abstract*—System Software is a fundamental core course for undergraduate students of Computer Science and Engineering. The traditional approach to teaching the System Software course within the School of Computer Science and Engineering lacked a meaningful connection to real-world machine architectures, leading to disinterest and reduced engagement among undergraduate students. This paper introduces an innovative teaching method designed to empower students to grasp the system programs of real systems effectively. Our approach involves effortlessly integrating the delivery of system software content with the Atmel AVR ATmega32 real-time machine, which students have previously encountered in a prior semester. Moreover, this paper provides a detailed examination of the use of a hypothetical machine in traditional teaching methodologies. While this method allowed for a more in-depth exploration of system software concepts, it struggled to establish a practical link to real machine. The novel teaching approach employed in this study adopts a unique method that links all the system software concepts with the practical system program of a real-time machine. This paper also explains how the advances in Technology has played a crucial role in considering real-time machines as examples. And it also discusses the limitations of teaching concepts using only hypothetical machine and concise overview of the chosen real-time machine is provided, followed by the observation of enhanced knowledge of system software concepts through its integration. To measure the effectiveness of the proposed methodology, we also gathered valuable feedback from the students. The course result analysis shows substantial improvement in understanding the concepts, performance and lifelong learning of the students.**

Keywords—System Software, Atmel AVR ATmega32, Hypothetical machine. Design thinking, Problem solving

I. INTRODUCTION

System Programming holds a crucial position within the undergraduate level computer science curriculum. The primary objective of the System Software (SS) course is to enhance students proficiency in system programming, a skill highly sought after by numerous leading industries.

JEET Category—Pedagogy of teaching and Learning
This paper was submitted for review on September 10, 2023. It was accepted on November, 15, 2023.
Corresponding author: Nagaratna D Kulenavar, School of Computer Science and Engineering, KLE Technological University, Karnataka, India
Address: Vidyanagar,Hubballi 580031 (e-mail: kulenavar@kletech.ac.in).

Traditionally, this subject has been taught using hypothetical computer systems as a foundation.

The existing system software syllabus has several limitations. First, it predominantly focuses on hypothetical machines, which are rarely employed in practice. Second, transitioning from hypothetical to real-time machines can be challenging. Third, analyzing real-time system software is complex, as it is developed for diverse real-time machines.

Lastly, the conventional teaching approach often employs hypothetical machines, despite the abundance of real-time machines in use. Therefore, to address these limitations, a technological shift is necessary.

The revised syllabus now includes an extensive case study highlighting the 32-bit ATMEL ATmega32 machine. This case study effectively addresses the shortcomings of the previous curriculum by actively engaging students in the development of system software for real-time machines currently in use. The outcomes of this study have demonstrated a creative impact on students and remarkably indicating a success rate exceeding 90%.

The course, along with its corresponding laboratory component, is introduced at the fifth semester level and carries a credit weightage of four. This comprehensive course structure includes 2 In-Semester Assessment (ISA), an End- Semester Assessment (ESA), and an extensive course project. Notably, the questions in both the ISA and ESA are carefully crafted to assess students' learning levels in alignment with Bloom's Taxonomy. The course, its delivery methods, and the evaluation techniques are all strategically designed to address the technical outcomes expected from the program.

In the context of teaching, the role of an instructor is pivotal in motivating and facilitating student understanding of the subject matter. Achieving this can be realized when instructors can provide effective solutions to the following critical questions:

Question 1: What innovative techniques can be introduced to dispel students' negative perceptions and encourage the application of suitable data structures for the implementation of system software algorithms?

Question 2: How can instructors guide students in the self- design of algorithms tailored for specific real machines, thus enhancing their grasp of the subject's core concepts?

To answer the above questions, we designed method that encouraged the application of suitable data structures in system software algorithms. We also employed innovative

techniques such as, interactive coding environments and real- world case studies which helped students connect theory with real-world applications and get a deeper appreciation for data structures. Furthermore, we guided students in self-designing algorithms for specific real machines by imparting a thorough understanding of machine architecture.

The paper is structured into several sections for coherent organization: Section II focuses on related work, Section III discusses conventional teaching methods, Section IV outlines the motivation behind choosing the ATmega32, Section V elaborates on the proposed teaching method, Section VI presents the results, and, finally, Section VII summarizes the conclusion and outlines paths for future work.

II. RELATED WORK

The paper Erdil (2020) showcases a pedagogical approach known as project-based learning, which prioritizes the practical application of computer architecture concepts through hands-on experiences. Instead of traditional lectures and theory-heavy coursework, students engage in projects and real-world problem-solving to gain a deeper understanding of computer architecture concepts.

The paper Rizki et al.,(2022) contributes to the understanding of the complex interplay between students' self-confidence and their mathematical problem-solving abilities, offering insights that can be valuable for educators and curriculum() designers in the field of mathematics education.

In the paper Hegade, P., Patil, N., & Kanthi, A. N (2022) the concept of creating a course portfolio in partnership with industry is explored, showcasing a forward-looking educational approach that offers advantages to both students and industry collaborators. It ensures that educational programs remain relevant and produce graduates who are well-prepared for the workforce, contributing to the growth and success of industries.The paper Husain, M., Tarannum, N., & Patil, N (2013).discusses novel teaching methods and strategies employed in the programming course elective to enhance the learning experience.

The book Beck (1985) "System Software: An Introduction to Systems Programming" is a comprehensive textbook that offers a thorough exploration of system software and its role in computer systems. Written by L. L. Beck, the book provides a valuable resource for students and professionals seeking to understand the foundations of systems programming. The paper Tingting Li & Zhan (2022) emphasis systematic review to critically evaluate and synthesize the available literature on the integration of design thinking into K-12 education. The aim is to provide insights into the impact, effectiveness, challenges, and best practices associated with incorporating design thinking principles.

The paper Patil, S. S., Dange, P., Pawar, A., & Patil, S. K (2022) discusses the "Design of a Hypothetical Machine with Complex Addressing Modes" project that offers a unique opportunity to enhance the understanding of system programming through simulation. By creating an interactive and educational platform, this project aims to empower learners and educators in the field of system programming, fostering a deeper appreciation for the intricacies of computer systems and memory management. In the paper Yunus et al, (2020) suggests Incorporating Design Thinking principles into the teaching of programming to gifted students can enhance their problem-solving abilities, creativity, and overall programming skills. It also helps them apply their programming Santos et al., (2020) knowledge to real-world scenarios, preparing them for future challenges in technology and innovation. The paper Hegade et al., (2021) exposes how courses provide students with real-world exposure and practical experience while benefiting industries through fresh perspectives and potential talent recruitment.

The paper Kulkarni, Nitya N., NirmalaPatil, K. G. Karibasappa, & MeenaMaralappanavar (2018) discusses how case study illustrates and how discrete mathematical structures play a crucial role in solving real-world problems, especially in complex systems. The paper Pörn et al., (2021) suggests that domain knowledge of programming is the most critical factor contributing to both a high perception of preparedness and a positive attitude. Teachers' perspectives on programming span a wide spectrum, ranging from a narrow focus on its association with elementary step-by-step thinking to more advanced reasoning that links programming to fundamental aspects of computational thinking and broader educational objectives. Authors Kelly & J. Gero (2021) encourages innovative solutions that are not only functionally robust but also genuinely meet the needs of the end-users. This integration of methodologies can be particularly valuable in fields such as user experience design, software development, and product design, where user satisfaction and efficient problem-solving are essential.

System software knowledge is a fundamental prerequisite for mastering system programming. Skill in system programming can be cultivated by exploring into subjects such as System Software, Compiler Design, Finite Automata, and Formal Language, among others. The System Software course includes the study of various machine architectures and the development of a deep understanding of Assemblers, Loaders, Linkers, and Macro processors. An assembler serves as a language translator, offering insights into the foundational translation processes that support nearly all programming languages, such as C, C++, Java, and more. Subsequently, for task execution to take place, programs must be loaded into memory, a vital function performed by system software known as the loader.

System software, including assemblers and loaders, plays a crucial role in computer science education. Computer science engineers must possess a thorough grasp of these elements. Therefore, the computer science curriculum(Sujatha, C., & Karibasappa, K. G (2015) incorporates system software to provide students with the necessary knowledge and skills related to assemblers, loaders, and essential system-related functions, equipping them for success in the practical application of their education.

III. CONVENTIONAL TEACHING

Traditionally, when teaching system software concepts as noted by Beck (1985), educators frequently employed hypothetical machines like the Simplified Instructional Computer (SIC) and Simplified Instructional Computer/extended (SIC/XE). These machines, along with simulators, were chosen over real assemblers such as Turbo Assembler (TASM), Microsoft Macro Assembler (MASM), Netwide Assembler (NASM), and others.

As every machine has its own architecture, hence every machine will have its own system software which needs to be explored by students. Whenever they will develop new system software definitely that machine will not be a hypothetical. But existing syllabus includes only hypothetical machine shown in Fig 1.

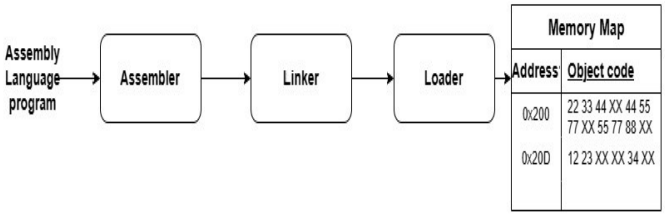


Fig 1: Flow of conventional teaching

Later hypothetical machine is replaced by real time machines to know practical and existing problems while designing and implementation of system software like assemblers, loaders, linkers and macro processor. First we explored with 8086 machine, then replacing 8086 with 8051 machine architecture, since 8051 architecture was studies in previous semester. Now, system software again is taught using hypothetical machine to illustrate all features like addressing modes Patil, S.S et al.,(2022) instruction formats, instruction set and input/output operations. Along with the case study of a real time machine to design and implement real time assemblers to unlock the power of real systems.

Limitations and Challenges of Conventional teaching:

1. Limitations:
   - Adaptation to other real time machine was difficult.
   - Facing obstacles while attempting to analyze the codebase of the existing system software.
   - Complexity in understanding other machine architectures.
   - Unable to practice assembly language programs on established assemblers like KEIL, MASM, TASM, Atmel AVR ATmega32, and others.
2. Challenges:
   - Time to learn existing machine architectural features.
   - Complexity in the design of pass1 and pass2 of two pass assembler algorithm for a real-time machine.

IV. REASONS FOR SELECTING ATMEL AVR ATMEGA32 MACHINE

In the existing syllabus system software is taught using two hypothetical machine architectural features to cover all concepts which are prerequisite to design any real time system software like assemblers, loaders, linkers and macro processor. Atmega32 is covered in Microcontroller course which is prerequisite for system software course. Also self-learning the architectural features for real time machine requires more time and writing of assembly language programs (ALP's) which are inputs to system software are machine dependent , so students need to write different ALP's to overcome this we need the course where the real time machine architectural features are taught, so that the students can directly start the design of system software, no need to learn architectural features because goal here is to design system software not to learn machine.

V. PROPOSED METHOD

In the past, computer systems were often viewed as black boxes, and people had limited insight into their inner workings. To facilitate the teaching of system software design and implementation, hypothetical machines with features resembling those found in real-world systems were used. This approach aimed to simplify and advance the learning process for system software development. This approach is highly effective as it enables students to systematically grasp essential concepts by working with model machines as part of their learning process. However, it's important to acknowledge that numerous real-time machines already exist, and students often study these architectures in various subjects such as Computer Organization as mentioned in paper Nayak et al.,(2021). Given the availability of these real machines, there's merit in teaching system software using a real-time machine. This approach allows students to address real-time problems in case studies and adapt to technological advancements effectively. Hence, recognizing the need for a technological shift is crucial. Hence, both hypothetical machines and real-time machines have their significance in the curriculum, as illustrated in Figure 2. This combination provides students with a well-rounded understanding of system software, encompassing both theoretical foundations and design of system software.

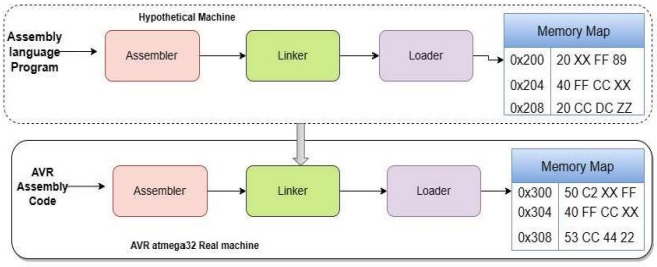


Fig 2: Flow of current teaching

Now, our proposed syllabus includes Atmel AVR ATmega32 real time machine as a case study to demonstrate design and development of system software like assemblers and loaders to illustrate the applying of knowledge in any real time machines.

The AVR machine is a low-power CMOS 8-bit microcontroller with 32 Kbytes In-System Programmable Flash Program memories with enhanced RISC architecture and optimizes power consumption versus processing speed. It

supports rich instruction set. The resulting architecture not only enhances code efficiency but also achieves throughput speeds that can be as much as ten times faster compared to traditional CISC microcontrollers. The Atmel AVR ATmega32 comes equipped with a comprehensive set of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Teaching system software through the proposed methodology has enabled students to gain a deep understanding of the subject within the system domain. This approach has motivated them to independently design system software for the Atmel AVR ATmega32 machine, ultimately empowering them to become expert System Programmers, as illustrated in Fig 3.

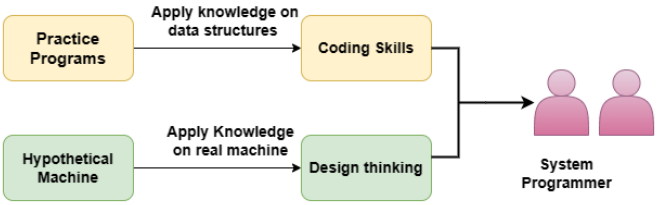


Fig 3: System software programmer requires the knowledge of coding skills and design thinking with team work

In our approach, students designed pass1 and pass2 of two assemble algorithm for Atmel AVR ATmega32 machine that are shown below

Algorithm 1: Pass1 of Two Pass Assembler

```
    Begin
      store #[OPERAND] as the starting address
      initialize LOCNCTR to the starting address
      append this line to the output file
      take the next line from ALP file
End {end for if ORG}
  else
    Set LOCNCTR to zero
  while OPNCODE != 'END' do
    Begin
      if this line is not a comment line then
        Begin
          if there is a symbol in the LABL field then
            Begin
              search SYMBOLTAB for LABL
              if LABL is found in SYMBOLTAB then
                Set an error flag
              else
                Add (LABL, LOCNCTR) into SYMBOLTAB
            End {end for if symbol}
          search OPNCODE in OPCTAB
          if OPNCODE is found in OPNTAB then
            Begin
              if OPNCODE is 'CALL' or 'JMP' or 'LDS' or 'STS'
                then
                Add 4 {instruction length} to LOCNCTR
              else
                Add 2 {instruction length} to LOCNCTR
            End
          else if OPNCODE is 'DB' or 'DD' or 'DQ' or 'DW' then
            Begin
              Compute the length and add it to LOCNCTR
            End {end for if BYTE}
          else if OPNCODE = 'BYTE' then
            Begin
              Compute the length in bytes and add it to LOCNCTR
            End {enf for if BYTE}
          else
            Error to indicate an invalid operation code
        End {end for if not a comment}
      append this line to the output file
      read the next line from ALP
    End {end for while not END}
  append the last sline to the output file
  Compute (LOCNCTR - starting address) as the length of the program
End {end of Pass 1}
```

The ability to design this type of algorithm in the field of computer science is a valuable skill for students and it can be applied in a wide range of projects across various fields. This design skill is a foundation of computer science, facilitating innovation, efficiency, and developing system programs like

- Operating System (OS The operating system (OS) serves as a foundational system program responsible for resource management and delivering essential services to support software applications.
- Device Drivers: Device drivers are essential system programs that enable the operating system to establish communication with and effectively manage hardware devices.
- Compiler: A compiler serves as a system program responsible for converting high-level programming languages like C and C++ into machine code, which is then capable of executionby the computer's central processing unit (CPU).
- Interpreter: Interpreters, which encompass programs such as Python interpreters and JavaScript engines within web browsers, are system programs designed to directly run high- level programming language code without the requirement of prior compilation.
- Linker: A linker serves as a system program responsible for combining object files, which contain compiled code, along with libraries, in order to produce a combined and executable program.
- Loader: A loader is responsible for loading executable programs into memory for execution.

Algorithm 2: Pass2 of Two Pass Assembler

```
Begin:
  read first line of the assembly program (from the output of Pass1)
  if OPNCODE = 'ORG' then
    Begin
      append to the output file
      read the next line of the assembly program
    End {end for if ORG}
  initialize ‘Text record’ and other necessary variables
  while OPNCODE != end_stmt do
    Begin
      if there are no ‘comment line’ then
        Begin
          check OPCTAB for OPNCODE
          if OPNCODE is present in OPCTAB then
            Begin
              if any symbol present in the OPERAND column then
```

```
                Begin
                  search SYMLTAB for OPERAND
                  if OPERAND is present in SYMLTAB then
                    set symbol address equal to operand address
                  else
                    Begin
                      set zero as address of operand
                      and set error flag
                    End
                End
              else
                set zero as address of operand
              gather the object code of instruction
              if the object code cannot be accommodated within
                  the existing 'Text record' space then
                Begin
                  append the 'Text record' to the object program
                  create a new 'Text record'
                End
              append the object code to the 'Text record'
            End {end for if opcode found}
          else if OPNCODE = 'BYTE' then
            convert the constant to object code
          append the listing line
        End {end for if not comment}
      read the next input line from ALP
    End {end for while not END}
  append the final 'Text record' to the object pgm
  append the 'End record' to the object pgm
  append the last listing line
End {Pass 2}
```

This methodology gives more practical knowledge, improves design thinking ability Vrana et al.,(2021), coding skills, easy to analyze existing system software code, adaptation from one machine to other machine, improve the quality of understanding level of students which assist them during their placement. The merging of Hypothetical and real time Atmel AVR ATmega32 machine for the design and development of system software worked very successfully till end of the semester. One sample ALP code which is input for the assembler is example for Hypothetical and Real machines is as follows in fig 4

Hypothetical Machine ALP

```
ADD   START 1000
      LDA NUM1
      ADD NUM2
      STA SUM

NUM1 WORD 32
NUM2 WORD 43
SUM   RESW 1
      END
```

ATmega32 Real Machine

```
ORG 0x1000
LDI R16,#32
LDI R17,#43
ADD R16,R17
STS 0x200,R16
END
```

Fig 4: Example of Hypothetical and Real machine input for assembler design

.

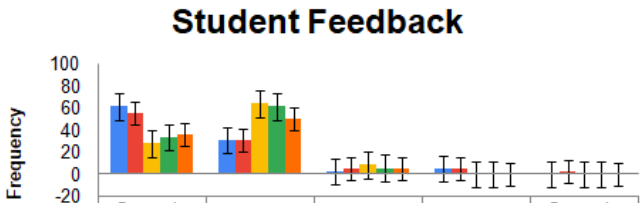
VI. RESULTS

The main goal of the present study was to explore the students' perspective on the design of real time system software and motivation as an important addition to experimental studies in the computer science education field. This exploratory case study found following evidences: Student Feedback: Students were asked the following questions as part of the feedback on the course delivery and importance.

The Questionnaire is as follows:

- Does System Software sound more interesting after had attended this course?
- Do you think that applying knowledge on real time machine in system software course is effective?
- Should the implementation of algorithms be done on real machine or hypothetical machine?
- Has case study approach improved your thinking and design skills?
- Have you accomplished the tasks effectively?

The conceptual mappings between the questions and categories are: questions 2 and 4 cater to challenge category; questions 1, 3 and 5 cater to responding phenomena category.



	Strongly Agree	Agree	Neutral	Disagree	Strongly disagree
1	61.11	30.55	2.77	5.55	0
2	55.55	30.55	5.55	5.55	2.77
3	27.77	63.88	8.33	0	0
4	33.33	61.11	5.55	0	0
5	36.11	50	5.55	0	0

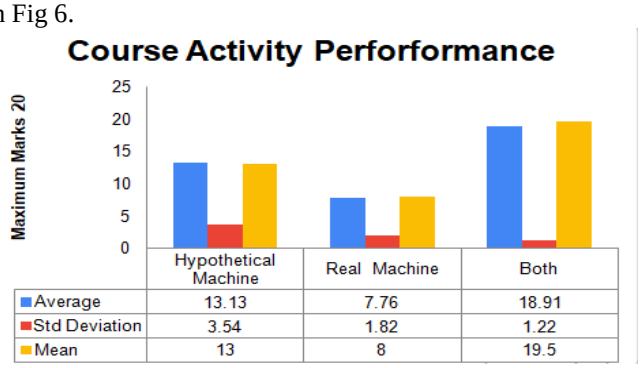
Fig 5: Plot of student feedback responses

The above graph showed in Fig 5 was the feedback given by students based on the questionnaire. Majority of the students strongly agree for their interest in the subject and applying of knowledge on real time machines based on questions 1 and 2. In the questions 3 and 4, students agree that there was an improvement in their design skills and implementation. . All students have completed the course project effectively that is evident through question 5. Based on the error graph, it is evident that there is room for improvement in the effectiveness of the method employed. This aspect will be carefully addressed and considered in our future work. Course Performance: Course project was carried out by making teams of 4 members. Each team was assigned either an assembler or a loader on 32-bit ATMEL ATmega32 machine. The below graph represents student's course project performance on three different attributes: Hypothetical machine, real machine and integration of both carried out in 3 different semesters and assessed for 20 marks. As it is evident from the statistical results, the third approach that is integration of both hypothetical and real

machine gave improved performance compared to the other two approaches.

Shettar et al., (2020) authors express importance of assessment of individual contributions in a team project can enhance team performance, promote fairness, and provide valuable insights for team members' growth and development. It's essential to strike a balance between quantitative data analysis and qualitative feedback to create a comprehensive assessment process. Individual contribution was assessed through the interaction with students in the course activity development.

The overall results of the proposed approach demonstrate that machine and adaptation has played a crucial role in promoting the comprehensive development of students, with the majority expressing high levels of satisfaction as shown in Fig 6.



	Hypothetical Machine	Real Machine	Both
Average	13.13	7.76	18.91
Std Deviation	3.54	1.82	1.22
Mean	13	8	19.5

Fig 6: Plot of student performance in the course project

VII. CONCLUSION AND FUTURE WORK

This research paper explored three distinct teaching approaches for the implementation of system software algorithms. These approaches resulted in the acquisition of practical knowledge, enhanced critical thinking skills, and facilitated the adaptation of existing software code from one machine to another. Overall, the proposed approach contributed to enriching the quality of teaching and the understanding level of students.

The proposed teaching methodology, which centered around real-time ATMEL AVR ATmega32 based system software, was introduced for undergraduate students. This approach effectively conveyed the significance of the system software course within the domain of system-side knowledge. As a result, which motivated the students to independently design assemblers and loaders, highlighting the practical importance of the subject matter.

The future directions for this research involve optimizing the proposed teaching methodology, potentially by incorporating parallel activities to enhance its efficiency within the given timeframe. Additionally, exploring more advanced processors beyond the Atmel AVR ATmega32 for the development of system software. There's also potential for integration with the Principles of Compiler Design course, allowing for the generation of ATMEL AVR Assembly Language Programs, which serve as input for the assembler, isanother aspect of this potential integration.

REFERENCES

Erdil, D. (2020). A project-based learning approach to teaching computer architecture. *EAI Endorsed Transactions on E-Learning*, *6*(19), 162289. doi:10.4108/eai.13-7-2018.162289

Rizki, N., Laila, A. R. N., Inganah, S., & armayanti, R. (2022).Analysis of Mathematic Connection Ability in Mathematics Problem Solving Reviewed from Student's Self-Confedence. In Seminar Nasional Teknologi Pembelajaran (Vol. 2, No. 1, pp. 111-126).

Hegade, P., Patil, N., & Kanthi, A. N. (2022). Building a course portfolio with industry- institute collaboration. *Journal of Engineering Education Transformations*,*36*(S1),50–55. doi:10.16920/jeet/2022/v36is1/22174

Husain, M., Tarannum, N., & Patil, N. (2013). Teaching programming course elective: A new teaching and learning experience. 2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE).

Beck, L. L. (1985). System Software: An Introduction toSystems Programming.

Li, T., & Zhan, Z. (2022). A systematic review on design thinking Integrated Learning in K-12 education. Applied Sciences, 12(16), 8077.

Patil, S. S., Dange, P., Pawar, A., & Patil, S. K. (2022). Design of new hypothetical machine of complex addressing mode for better understanding of system programming through simulation.

Prakash Hegade, P., Patil, N. S., & Bidari, I. (2021). Principles of Elective Design with Industry-Institute Collaboration. Journal of Electrical Engineering & Technology, 34, 384. https://doi.org/10.16920/jeet/2021/v34i0/157184

Balakrishnan, B. (2021). Exploring the impact of design thinking tool among design undergraduates: A study on creative skills and motivation to think creatively. *International Journal of Technology and Design Education*, 1–14. https://doi.org/10.1007/s10798-021- 09652-y

J. Liedtka & Marcos Chin. (2022). Why Design Thinking Works.Meinel, C., & Leifer, L. (2012). Design Thinking Research. Understanding Innovation, 1–11. https://doi.org/10.1007/978-3-642- 21643-5_1

Shettar, A., Nayak, A. S., & Shettar, A. (2020). Assessing individual contribution in a team project using

Learning Analytics.Procedia Computer Science, 172, 1001-1006.

Nick Kelly & J. Gero. (2021). Design thinking and computational thinking: A dual process model for addressing design problems. Design Science. https://doi.org/10.1017/dsj.2021.7

Pörn, R., Hemmi, K., & Kallio-Kujala, P. (2021). Inspiring or confusing – a study of Finnish 1–6 teachers' relation to teaching programming. 9(1), 366-396. https://doi.org/10.31129/lumat.9.1.1355

Santos, S. C. dos, Tedesco, P., Borba, M., & Brito, M. (2020). Innovative Approaches in Teaching Programming: A Systematic Literature Review. International Conference on Computer Supported Education,205–214. https://doi.org/10.5220/0009190502050214

Nayak, A. S., Hiremath, N. D., Umadevi, F. M., & Garagad, V. G. (2021). A hands-on approach in teaching computer organization & architecture through project based learning. Journal of Engineering Education Transformations, 34, 742-746

.

Sujatha, C., & Karibasappa, K. G. (2015). Curriculum Design: An Experience in Principles of Complier Design Course. Journal of Engineering Education Transformations, 28(Special Issue).

Vrana, J., Johannes Vrana, Johannes Vrana, & Singh, R. (2021).NDE 4.0—A Design Thinking Perspective. Journal of Nondestructive Evaluation, 40(1), 1–24. https://doi.org/10.1007/s10921-020-00735-9