From Configuration to Cognition: A Computational Thinking Approach to Computing Infrastructure Education

[1]S. Thiruchadai Pandeeswari, [2]G. Vennila
[1], [2]Department of Information Technology, Thiagarajar College of Engineering, Madurai-15
[1] eshwarimsp@tce.edu, [2]gvait@tce.edu

Abstract— **AI innovations have led to unprecedented demand and growth in computing infrastructure domain, creating promising avenue of employment for role-ready Engineering graduates. In this context, computing infrastructure studies garner special focus. Though there are several studies on integrating CT skills in K-12 and Engineering Education, thre are few to none on integrating CT skills into Computing Infrastructure education. This study aims at proposing a unified framework for integration of Computational Thinking (CT) skills into the course design, delivery and assessment of computing infrastructure studies targeting undergraduate Engineering students of IT discipline. The proposed approach is implemented for a foundational course on IT Infrastructure, with a focus on enhancing student cognition and problem-solving capabilities. A structured pedagogical approach is employed to map course activities to core CT components—Decomposition, Pattern Recognition, Abstraction, and Algorithmic Thinking—using Bloom's taxonomy for skill-level alignment. A carefully planned Assessment framework to evaluate the attainment of CT skills through guided activities is also proposed. Quantitative evaluation of the proposed approach was conducted by considering the performance of 120 students across eight assessment items, with correlation mapping and Cronbach's alpha used to validate the reliability of the skill measurement. Results indicated strong correlations among Pattern Recognition, Abstraction, and Algorithmic Thinking, with Cronbach's alpha values for Decomposition and Algorithmic Thinking approaching the benchmark of 0.7, confirming acceptable consistency. Performance distributions showed high class averages, narrow interquartile ranges, and minimal outliers, suggesting effective learning outcomes. Perception-based evaluation using a Likert-scale survey revealed uniformly positive student feedback, with average ratings between 4.27 and 4.54, indicating a strong sense of CT skill acquisition. The findings demonstrate that integrating CT skills into course pedagogy can significantly improve technical proficiency, cognitive engagement, and student confidence in solving computing infrastructure-related problems.**

Keywords— **Computational Thinking, Algorithmic Thinking, IT Infrastructure, Computing Infrastructure, Infrastructure Studies.**

ICTIEE* Track*: Emerging Technologies and Future Skills
ICTIEE Sub-Track: Preparing Engineers for Digital and Sustainable world details.)

I. INTRODUCTION

ARTIFICIAL Intelligence innovations accelerated an unprecedented expansion of computing infrastructure around the world. The expansion is estimated to grow by 160% by 2030 (Sachs, G. (2024)). The AI infrastructure market is expected to reach a $ 395 billion market by 2030 (Report by Markets and Markets). The increased adoption of AI necessitates a robust education framework for computing infrastructure. Computing infrastructure studies have become more pertinent with these advancements in AI. As AI-enabled systems demand scalable and resilient high- performance architectures, the focus on infrastructure design and optimization has become critical. While AI automation may reduce certain human roles within computer engineering, the evolving infrastructure landscape promises to generate new employment avenues, making it a vital growth domain for the discipline. Thus, it is crucial to effectively integrate computing infrastructure studies in the undergraduate Computer Science Engineering and allied disciplines.

Computing Infrastructure study involves analyzing and facilitating hardware, software and networking resources for Computing operations and services. Acquiring skills relevant to designing and analyzing computing infrastructure would enable the students of Undergraduate Computer Science Engineering and allied disciplines to design scalable and efficient systems that meet business and application-specific needs. In this paper, a systematic approach for incorporating computational thinking pedagogy into courses that deal with computing infrastructure studies is implemented and empirically validated.

Computational Thinking (CT), a term introduced by Jeanette Wing (Wing, J.M 2006), describes a way humans approach solving problems by drawing on concepts fundamental to computer science. Wing emphasized that CT is an essential skill for everyone as it is integral to analytical ability. Computational thinking encompasses the four essential thought processes of decomposition, pattern recognition, abstraction and algorithmic thinking (Selby and Woollard 2013). These cognitive processes are the foundation for formulating problems and deriving their solutions. Computational thinking is usually associated with problem solving and hence widely adopted for programming courses in the Computer Science Engineering curriculum. Courses that focus on computing infrastructure, such as System Administration, Network Administration and DevOps, mostly focus on rote learning with the set up procedures and commands for tools usage. Integrating Computational Thinking skills into the computing infrastructure-related courses in the curriculum offers significant benefits in moving learners from rote memorization to a more dynamic problem-solving approach. The benefits include:

1. Promotes deeper conceptual understanding, enabling students to think beyond tools and commands
2. Allows a problem-solving approach towards infrastructure studies by converting tasks into open ended problems
3. Promotes abstraction, modelling and automation
4. Facilitates automation scripts, orchestration of workflows and proactive monitoring and response systems, especially in the context of cloud-native and AI-driven architectures.
5. Emerging fields like CI/CD pipelining, AI-aided operations demand CT-driven reasoning

However, facilitating the integration of computational thinking pedagogy into computing infrastructure education has significant challenges due to the inherent nature of courses dealing with computing infrastructure studies. The major challenges are as follows:

1. Systematic approaches to infusing CT concepts such as decomposition, pattern recognition, abstraction, and algorithmic thinking into CI courses is very limited.
2. Embedding CT into already content-rich CI courses requires careful planning, else it may overwhelm the learners
3. Integrating CT into CI courses requires professional development to understand how to integrate abstraction, algorithms, and systems thinking into CI topics.
4. Lack of rubrics or tools to assess how well CT has been applied in CI tasks like building a virtualized environment or configuring a cloud network.

Considering the benefits and challenges in integrating CT-based pedagogy into CI courses, this paper explores the solution approaches to overcome the aforementioned challenges. This paper presents an empirical study on the inclusion of CT skills into a computing infrastructure course. To explore the underlying dynamics of integrating computational thinking pedagogy into computing infrastructure courses, this paper is structured around the following research questions.

1. How can computational thinking skills be systematically embedded into a computing infrastructure course to enhance students' problem-solving abilities beyond procedural task execution?
2. What assessment frameworks can effectively measure students' application of computational thinking skills, such as decomposition, abstraction, and algorithmic problem-solving?

Even though CT has been widely integrated into the programming and K-12 STEM programs, very little has been done to apply it to computing infrastructure courses. The current CT models lack the cognitive mechanisms that would be applied in activities like architectural design, storage layout, Information Lifecycle Management (ILM) processes, and enterprise network analysis. This work consequently introduces a new CT-based pedagogic model, sensitive to computing infrastructure teaching and learning and based upon a statistically verified evaluation model. It is the initial reported case where CT skills have been systematically mapped to IT infrastructure activities and evaluated in terms of reliability by means of statistical validation of results with student performance data. The main significance of the presented research is in the systematic incorporation of the main elements of CT, namely Decomposition, Pattern Recognition, Abstraction, and Algorithmic Thinking, into the instructional design framework, delivery model, and evaluation of a 4-year IT Infrastructure program. This teaching hypothesis has undergone empirical approval on a group of 120 students, and can provide data about enhanced cognitive and problem-solving skills unique to infrastructure training.

II. Related Works

This section organizes prior work relevant to integrating CT pedagogy into undergraduate computing-infrastructure courses, categorized into four themes: data-driven approaches, teacher-focused studies, multi-method perspectives, and pedagogical innovations with real-world alignment.

A. Data-driven Method

A data-driven perspective of large-scale case study on learner-centered feedback analytics in higher education, revealed how nuanced feedback patterns can inform instructional design, improve learner engagement, and tailor content delivery (Aldino et al. 2022). Wang et al. (2021) conducted a scoping review on the interplay between CT and creativity, identifying shared cognitive foundations such as abstraction, iterative refinement, and divergent thinking, while highlighting opportunities for cross-skill development in computing curricula. Similarly, Lin et al. (2023) investigated learner-centred analytics of feedback content, offering actionable insights into improving educational interventions through data analysis, adaptive scaffolding, and personalized learning pathways.

Recent work has expanded this data-driven perspective by integrating advanced analytics, visualization, and adaptive systems into computing education. Johnston et al. (2024) conducted a systematic review of learning analytics in computing education, identifying emerging frameworks for feedback personalization and skill tracking. Lee and Kim (2024) designed a self-determination theory-informed learning analytics dashboard, demonstrating increased student motivation and participation in asynchronous learning environments. Huang et al. (2025) examined the usability and visual design of learning analytics dashboards in health professions education, emphasizing their applicability in skill-based computing courses.

Kaliisa et al. (2024) explored how students engage with analytics feedback in higher education, proposing design principles to foster self-regulated learning. Johnston et al. (2024) applied interpretable machine learning models to predict student engagement in computing modules, offering transparent and actionable predictions. Sadallah (2025) introduced an adaptive understanding framework for learner-centered dashboards, enhancing personalization and learning support. Adeyemi and AlOtaibi (2025) developed a real-time feedback decision support system using Light Gradient Boosting Machine (LightGBM) and SHapley Additive exPlanations (SHAP) explainable AI, ensuring both adaptivity and interpretability in feedback delivery. Zhou et al. (2025) proposed a tag-based automated feedback generation approach using ChatGPT, validated through teacher evaluations for its educational relevance and accuracy.

B. Focusing on Teachers

Holstein and Cohen (2025) examined how Scratch educators perceive integrating CT into other school subjects within a constructionist framework. Teachers noted that CT not only cultivates computational skill-building but also enhances creativity, collaboration, and subject relevance, especially in mathematics and science contexts. Hirt et al. (2025) evaluated a professional development (PD) program aimed at strengthening teachers' self-regulated learning competencies. Their findings showed measurable gains in instructional planning, reflective practice, and adaptive decision-making. Liu et al. (2024) offered a systematic review of K–12 teachers' PD for CT instruction, spotlighting challenges like limited curriculum time, diverse expertise levels, and resource constraints, while identifying best practices such as mentoring and collaborative curriculum design. Rodrigues et al. (2024) conducted a systematic review on integrating CT into initial teacher training for primary schools. They concluded that effective CT education requires long-term, comprehensive training that blends both theoretical foundations and practical, reflective components. Greifenstein et al. (2023) explored how primary school teachers in training design programming tasks. They found that tools like LitterBox, which provides real-time feedback on code quality, helped pre-service teachers move from thematic brainstorming to objective-driven task creation, reducing misconceptions and improving task quality

C. Multi-Method in CT Education

From a multi-method perspective, Varela et al. (2023) assessed CT skills in engineering and computer science students using a mixed-methods design. This allowed for a comprehensive evaluation of both conceptual understanding and practical application, illuminating how students display CT differently in project-based versus theoretical contexts. Ali and Smith (2023) examined cross-case CT implementations in K–12 subjects, revealing interdisciplinary benefits like enhanced problem-solving fluency and challenges such as aligning assessments and ensuring curricular coherence. Shah et al. (2024) conducted a systematic review of CT professional development initiatives, finding that blended learning models were particularly effective for long-term teacher retention. Wu and Li (2024) proposed a holistic framework positioning CT as a data-centric literacy, arguing that this framing supports equity by developing critical engagement with data-driven decision-making. Yeni et al. (2024) conducted a *systematic review* of 108 peer-reviewed studies on interdisciplinary CT integration in K–12. Their analysis mapped subject areas, instructional strategies (like block-based tools), and research designs, revealing that most implementations focus on science and math, often at a low substitution level rather than transformative integration.

Falloon (2024) deployed a *structured, problem-based curriculum* for early years (approx. age 6), using floor robots to assess CT development (e.g., algorithm writing, pattern recognition). He demonstrated that young students can achieve both foundational and higher-order CT skills through guided inquiry pedagogy.

Subramaniam et al. (2025) applied a STEM Ways of Thinking framework in an undergraduate physics course, analyzing student-generated engineering design-based problems. This qualitative, design-based study used transcript and report analysis to explore how students incorporate CT (via Python coding), metacognition, and iterative problem framing. Adorni et al. (2024) presented the FADE-CTP framework, a mixed-methods design framework for analyzing educational CT problems (CTPs). The approach involves systematically profiling tasks' characteristics and the CT competencies they activate, aiding both assessment design and teacher training. Table I illustrates how diverse research designs from structured curricula to qualitative task analyses and systematic reviews provide nuanced insights into CT integration across educational contexts.

D. Software Engineering Education: Pedagogical Innovations and Real-World Alignment

Haugen and Stålhane (2022) identified persistent challenges in DevOps instruction namely tool complexity, cultural shifts in team collaboration, and difficulties assessing distributed workflows. Building on this, Haugen (2022) proposed project-based and collaborative learning models that significantly enhanced both students' technical skills and teamwork abilities, while aligning classroom practice with realistic industry workflows to increase employability and foster lifelong learning habits. Complementing DevOps-focused studies, Gransbury et al. (2023) designed a project-based software engineering curriculum for secondary students, incorporating block-based programming environments, APIs, and socially relevant themes. Their nine-week module yielded strong student engagement, especially in autonomy, creativity, and collaborative problem-solving. Afshar et al. (2022) presented an integrated software engineering curriculum through Project-Based Learning [PBL] at the undergraduate level. The program emphasized real-world engineering challenges and fostered both technical proficiency and soft skills.

Garces and Oliveira’s (2024) four-year experience report on PBL in software engineering courses revealed recurring themes in design, collaboration, and iterative refinement, offering educators actionable insights on methods that worked well versus those needing revision. In the realm of DevOps tool education, Iyer et al. (2024) introduced a web-based IDE tailored for DevOps learning in higher education. It enabled cloud-based, accessible environments with tutorials and automated setup, catering to both students and practitioners. Moreover, Garcia et al.’s (2024) systematic literature review of DevOps teaching techniques highlighted that active, student-centred methods—especially PBL, collaborative learning, and flipped classrooms—were predominant and effective, although validation of student performance and curriculum currency remains challenging. From another angle, Borja-Fernández et al. (2023) proposed an automatic assessment framework for team coding assignments in DevOps contexts. By leveraging version control metrics, they achieved fair distribution of individual performance and transparent feedback—enhancing trust and motivation in team-based software projects. Moving beyond collaboration, Bonetti et al. (2025) conducted a systematic mapping study on Example-Based Learning (EBL) in software engineering education. Their findings show that EBL not only boosts student motivation and conceptual understanding but also supports applying theoretical concepts through concrete examples—though instructor effort and resource constraints remain barriers.

TABLE I
SYSTEMATIC REVIEWS WITH CT INTEGRATION

Study	Methodology	Focus
Varela et al. [19]	Mixed methods	CT in project vs theoretical contexts
Yeni et al. [29]	Systematic review	Interdisciplinary CT studies in K–12
Falloon [30]	Structured curriculum study	CT development in early years
Subramaniam et al.	Qualitative STEM design tasks	CT in physics with Python
Adorni et al. [32]	Framework development	Analytical profiling of CT tasks

Even though CT has been widely integrated into the programming and K-12 STEM programs, very little has been done to apply it to computing infrastructure courses. The current CT models lack the cognitive mechanisms that would be applied in learning activities like architectural design, storage layout, Information Lifecycle Management (ILM) processes, and enterprise network analysis. This paper introduces a new CT-based pedagogic model, sensitive to computing infrastructure teaching and learning.

III. METHODOLOGY

A. Key Contributions

In this work, CT skills are systematically mapped to the Teaching-learning activities for computing infrastructure education and statistically validated. This work focuses on proposing an unified course design, delivery and assessment framework for integrating CT skills into computing infrastructure education. The key contributions are as follows:

- Framework for mapping Computing Infrastructure course contents and CT Skills
- Assessment Framework for evaluating CT Skills in computing infrastructure courses.
- Empirical validation of the proposed framework statistical evaluation and student perception based qualitative analysis.

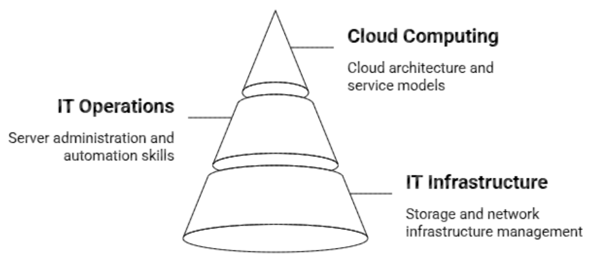

Cloud Computing
Cloud architecture and service models
IT Operations
Server administration and automation skills
IT Infrastructure
Storage and network infrastructure management

Fig. 1. Three-layered computing Infrastructure Education

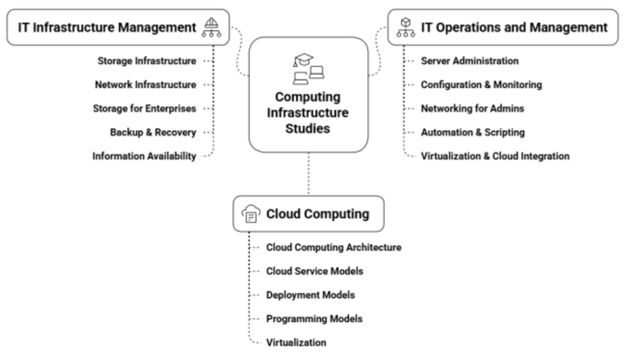


Fig. 2. Key Areas of Computing Infrastructure Studies

The main significance of the presented research is in the systematic incorporation of the main elements of CT, namely Decomposition, Pattern Recognition, Abstraction, and Algorithmic Thinking, into the instructional design framework, delivery model, and assessment of courses imparting computing infrastructure education.

B. Three-Layered Computing Infrastructure Education Stack

To include computing infrastructure studies in the undergraduate Engineering discipline of Information Technology, a three-course structure is chosen as shown in figure 1. The foundation for computing infrastructure is placed at the II semester, when programming is also introduced to the students. The foundation course is followed by the System Administration course, a course that deals with Server administration commands and tools in the III semester and the Cloud computing course, a course that deals with modern, scalable cloud computing systems in the V semester.

The IT Infrastructure Management course introduces students to enterprise-grade data storage technologies, networked storage models, and real-world storage networking systems. The course aims to equip them to manage and maintain critical data and ensure business continuity through hands-on exposure to storage solutions. The System Administration course strengthens students' foundation in administering Linux and Windows servers, including configuring and monitoring key server roles, and extends their skills with practical shell scripting. This enables students to acquire skills relevant to effective automation and management of IT resources.

The third course on Cloud Computing expands students' knowledge to distributed and service computing, focusing on cloud architecture, service models, deployment strategies, and resource scheduling. It prepares them to design, deploy, and manage scalable cloud-based solutions that meet dynamic enterprise demands. Figure 2 shows the key areas of computing Infrastructure Studies in undergraduate Engineering discipline

This three-course structure provides a layered competency-building approach, starting from hardware & storage foundations, moving to operational management, and culminating in distributed/cloud systems. Together, these courses foster key computational thinking skills such as decomposition, algorithmic thinking, pattern recognition, and abstraction, allowing students to systematically analyze, design, and implement complex IT infrastructures. This integrated curriculum ensures graduates are well-prepared with both theoretical foundations and practical expertise to address modern challenges in computing infrastructure, enterprise data management, and cloud technology. It is aimed to impart all the three courses through CT skills-based pedagogy. As a pilot study, CT skills-based content design, delivery and assessment were experimented with in the course on IT Infrastructure Management.

C. RQ1: Embedding CT into Computing Infrastructure Course Design

The course on IT Infrastructure Management aims to lay a strong foundation on the computing infrastructural elements required for deploying enterprise-grade applications. The course is designed for students of the undergraduate B.Tech IT discipline. The key areas covered by this course are illustrated in figure 3 below.

The objective is to impart CT skills by aligning them with the course contents systematically. The following four CT skills are predominantly considered for mapping with the course contents.

- Decomposition: ability to break down complex storage devices and technologies into smaller sub-components and understand them better.
- Pattern Recognition: ability to identify similarities or common characteristics within infrastructural requirements and challenges for supporting applications. This helps them to evolve better infrastructure solutions.
- Abstraction: the ability to filter low-level details and prepare a high-level perspective of the concepts.
- Algorithmic Thinking : ability to develop step-by-step solution to a infrastructural problem by applying appropriate formula / logic / order of operations.

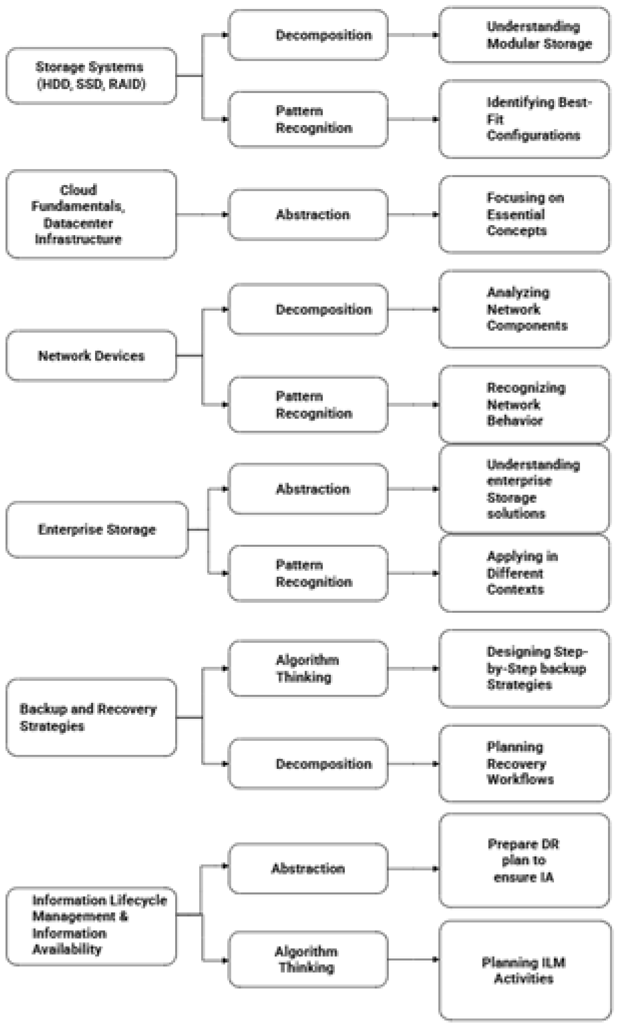

Storage Systems (HDD, SSD, RAID)
Decomposition
Understanding Modular Storage
Pattern Recognition
Identifying Best-Fit Configurations
Cloud Fundamentals, Datacenter Infrastructure
Abstraction
Focusing on Essential Concepts
Network Devices
Decomposition
Analyzing Network Components
Pattern Recognition
Recognizing Network Behavior
Enterprise Storage
Abstraction
Understanding enterprise Storage solutions
Pattern Recognition
Applying in Different Contexts
Backup and Recovery Strategies
Algorithm Thinking
Designing Step-by-Step backup Strategies
Decomposition
Planning Recovery Workflows
Information Lifecycle Management & Information Availability
Abstraction
Prepare DR plan to ensure IA
Algorithm Thinking
Planning ILM Activities

Fig. 4. Mapping of the course contents to the CT skills

The mapping has been carried out by matching the inherent nature of the topic and the CT skills. Also, the cognitive level at which the students need to learn the concepts is also considered. Theoretical concepts such as parts of HDD, SSD, network components, IA metrics are dealt at 'Understand' level (K2) of Blooms taxonomy. The CT skill predominantly focussed for concepts at K2 level is 'Decomposition'. Similarly, practical concepts such as Backup and Disaster recovery planning, planning infrastructure for enterprise applications are dealt at 'Apply' (K3) level of Bloom's taxonomy. Such topics are predominantly delivered in alignment with the CT skills – abstraction and algorithmic thinking, which will facilitate the learners to chart out effective plans. The mapping of the course contents to the CT skills is illustrated in figure 4.

TABLE II
ASSESSMENT FRAMEWORK TO EVALUATE CT SKILLS

Activity	CT Skill addressed	Description	Guidelines	Deliverable
A1.a	Decomposition	Breaking down components of Hard disk drive and understanding each of the component's function.	A demonstration video that contains explanation of physical sub-components of a HDD	A handwritten report summarizing the working of HDD
A1.b	Decomposition	Breaking down the network settings on a personal computer and learning the network properties such as protocols, IP address, MAC address etc.,	A document containing necessary steps/commands to check network settings	A document containing screenshots of the steps carried out and the current network setting of their computer.
A2	Decomposition	Identify the necessary infrastructural elements required for deployment of a enterprise application with specific demands	Scenario-based problem statements set	Identification of core infrastructural elements such as network, storage, backup etc based on the requirements of the problem statement chosen.
A3	Pattern Recognition	Enterprise Networking Case Study	Reference websites (Cisco, Juniper Case Studies)	Report based on a chosen white paper, explaining the networking problem and the solution proposed at enterprise level
A4	Abstraction	Mindmap on Datacenter – Components, need, characteristics	Virtual tour of Google, Microsoft Datacenters	A mindmap that reflects all the essential concepts about Datacenter
A5	Abstraction	Learn the components of SSD and the differences with HDD.	Use AI to learn	Frame 20 curious questions and answers about SSD and how it is different from HDD.
A6	Abstraction	Derive a high-level architecture of infrastructure required for deployment of an enterprise application	Scenario-based problem statements	Report containing high-level architecture diagram for the chosen enterprise application with specific requirements
A7	Algorithmic Thinking	Solving problems given in a worksheet	Sample worked out problems	Solution to the problems by applying appropriate logic/ formula/ sequence of steps.
A8	Algorithmic Thinking	Perform Information Lifecycle Management activities for a chosen application and derive a back up plan	Flow of ILM and Backup considerations	Backup plan for a chosen problem statement.

D. .RQ2: *Assessment frameworks for measuring attainment of CT skills by students*

To effectively assess the attainment of computational skills by students, a set of eight key activities that inherently promotes the CT skills was identified. All these activities were given as hands-on activities to students in a guided format. Students will be asked to follow the guidelines given, perform the learning activity and expected to produce a well-defined deliverable. As highlighted in the above section, the activities mapped to the "Decomposition" skill are evaluated at K2 level. The activities mapped to 'abstraction' and 'algorithmic thinking' are evaluated at K3 level of Blooms taxonomy.

The description of the activities, details of the guidelines given to the students to carry out the activities and the deliverable expected from the students are listed in the Table II above. The students were asked to carry out the activities in lab as soon as the relevant theory concepts are discussed in the class. This allowed them to apply the CT skills learnt in solving the problems given.

IV. RESULTS AND DISCUSSIONS

A. *Quantitative Evaluation*

The effectiveness of integrating CT skills into the Content design and delivery of the infrastructure management course is measured through quantitative evaluation and perception-based evaluation. For quantitative performance evaluation, the performance of students in the set of eight activities is considered. The assessment items were attempted by 120 students.

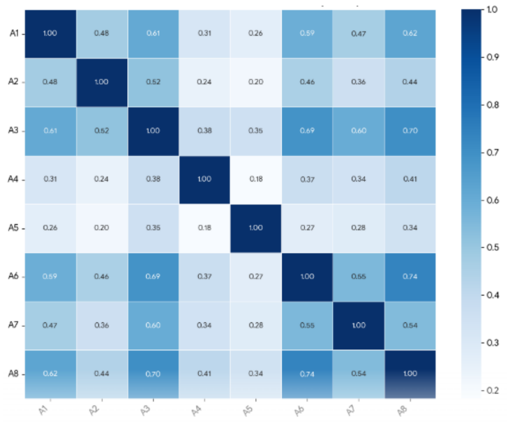


Fig. 5. Correlation Map of eight activities

The skill mapping of the assessment items are as follows:

1. Decomposition → A1, A2 (Max marks: 8 each, Bloom's K2 level)
2. Pattern Recognition → A3 (Max marks: 16, Bloom's K3 level)
3. Abstraction → A4, A5, A6 (Max marks: 4, 4, 16, Bloom's K3 level)
4. Algorithmic Thinking → A7, A8 (Max marks: 8, 16, Bloom's K3 level)

The correlation map of all eight activities is charted to understand how the activities are interrelated to one another and the causality among them. The correlation map illustrated in figure 5 shows strong correlation among the items A3, A6, A7, A8, which assess Pattern Recognition, Abstraction, and Algorithmic Thinking. The strong correlation indicates the important fact that the students were able to identify patterns, abstract the details and propose a solution by applying algorithmic thinking.

Further, to ensure the correlation among the related components and the internal consistency of the assessment data, Cronbach's alpha value was calculated for (A1, A2), (A4, A5, A6) and (A7, A8). The comparison of the cronbach's alpha of the related items and the benchmark value of 0.7 is shown in figure 6 below. The results show that the activities used to measure the CT skills, Decomposition and algorithmic thinking have acceptable consistency with values approaching the benchmark value 7. However, the assessment items used for measuring the CT skill, 'abstraction' show poor internal consistency. Both the correlation map and the measured Cronbach's alpha values ensure the correctness and validity of the assessment items chosen for performance evaluation. Further, the overall distribution of the marks scored by students illustrated in figure 7 shows a left-skewed bell curve, which strongly suggests a positive outcome with most students scoring high. The same trend is confirmed by the box-plot visualization shown in figure 8.

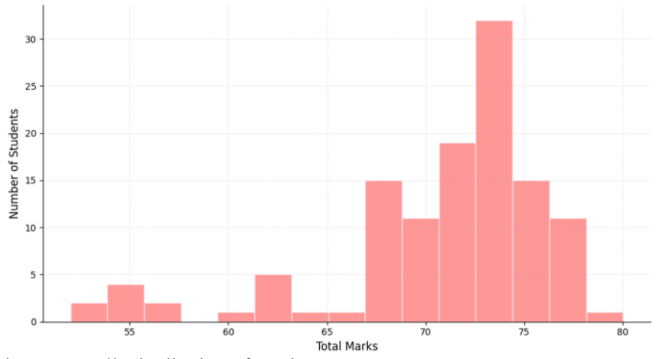


Fig. 7. Overall Distribution of Marks

The assessment items A1, A2 show small interquartile range

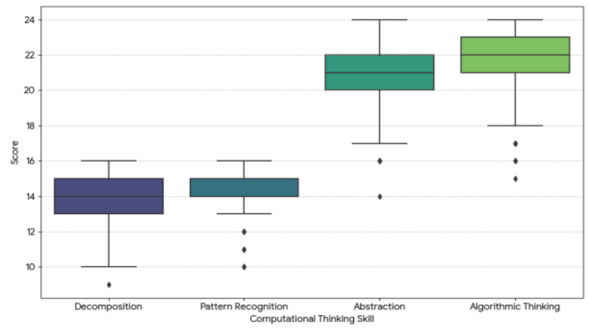


Fig. 9. CT Skillwise distribution of student scores

indicating good performance by most of the students at K2 level. A3 shows slightly wider distribution than (A1, A2) and around 10% of students scoring significantly lower. The (A4,A5,A6) set meant to measure Abstraction shows consistent performance by students. Similar good and consistent performance observed in (A7,A8) measuring algorithmic thinking. Presence of few outliers indicate 7-10% of students struggling to adapt. For better clarity, CT skillwise distribution of marks is shown in figure 9. The average marks scored in each assessment item is shown in figure 10.

The results demonstrate that, except few outliers, majority of the students were able to acquire CT skills and hence achieve expected level of cognition. The high overall class averages and the tight interquartile range indicate the significant effect of the CT skills-based pedagogy followed in the IT Infrastructure management course.

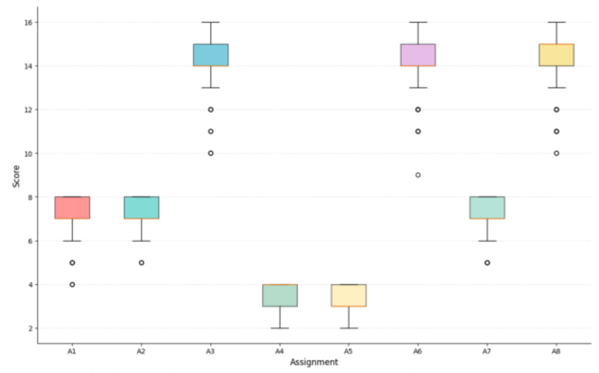


Fig. 8. Distribution of student scores across eight assignments (A1–A8) mapped to computational thinking skills

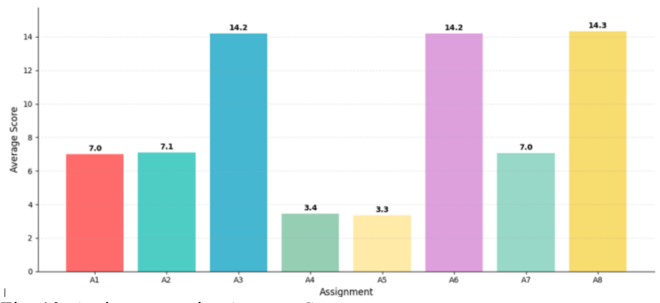


Fig. 10. Assignment-wise Average Score

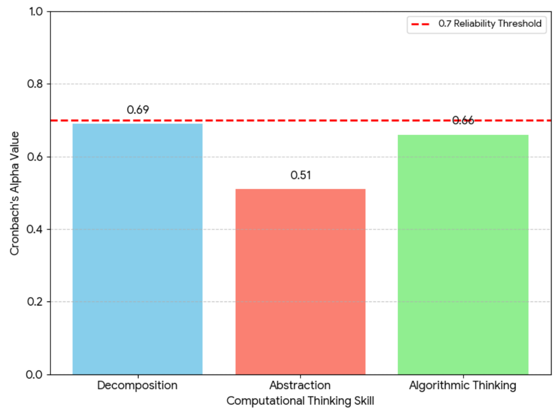


Fig. 6. Comparison of internal consistency of assessment activities measured through Cronbach's alpha

Further, the strong correlation among Pattern Recognition, Abstraction, and Algorithmic Thinking aligns with Wing's view (Wing, J.M 2006) of Computational Thinking as a network of cognitive mechanisms that support both problem formulation and solution generation. The skill mapping is in line with the tripartite CT model of Selby and Woollard conceptual, algorithmic, and developmental layers. The development between Decomposition (K2) and Abstraction and Algorithmic Thinking (K3) supports the revised taxonomy by Bloom, which means one is involved in higher-order cognitive activities.

About 7-10 percent showed problems with tasks that are abstraction-intensive (A4-A6) which implies that they had problems in shifting towards conceptual modeling of learning, rather than the previous course of procedural learning. Other students had difficulty in comprehending enterprise level situations especially when asked to create high level infrastructure architectures. Some of the students needed further assistance in form of scaffolding when performing practical exercises because they were not familiar with storage and networking systems. These issues highlight the need to have instructional differentiation and specific support systems.

B. Perception-based evaluation

The effectiveness of the CT skills based pedagogy for computing infrastructure courses is further measured through perception-based evaluation. To carry out the perception-based evaluation, a short survey questionnaire depicted in the Table III is used.

TABLE III
PERCEPTION-BASED EVALUATION SURVEY QUESTIONNAIRE

Q.No	Questions	CT Skill
1	The course activities and assignments helped me to break down IT Infrastructure into storage and network components and understand them better	Decomposition
2	The activities given effectively helped me to break down complex infrastructure structures	Decomposition
3	The course content helped me to learn the recurring patterns, requirements in IT Infrastructure	Pattern Recognition
4	The activities helped me break a problem into sub-problems and identify solutions by applying appropriate logic	Algorithmic Thinking
5	The activities helped me to consolidate low-level details of the Network, storage and propose a high-level architecture for applications	Abstraction
6	Please provide specific examples of how you have applied any of the skills learnt from the course to solve a problem, either inside or outside of this course.	-

The average Likert scale scores (1-Strongly disagree, 5-Strongly agree) presented in figure 11 indicates the strong sense of attainment of CT skills among the students. The range of the average score (4.27-4.54) indicates uniformly positive perception on the acquisition of CT skills. Further, some of the impactful Examples of the application of skills learnt from the course given by students are illustrated in figure 12.

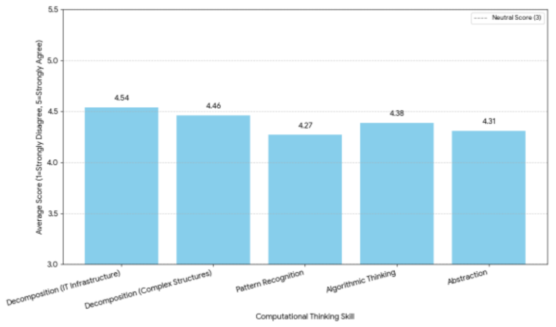
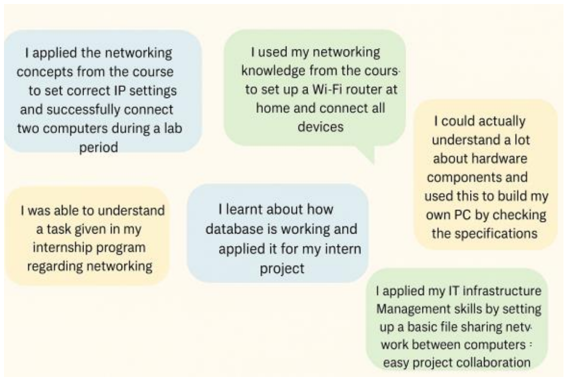


Fig. 11. Average Likert scale scores (1 = Strongly Disagree, 5 = Strongly Agree) from perception-based survey



I applied the networking concepts from the course to set correct IP settings and successfully connect two computers during a lab period

I used my networking knowledge from the cours to set up a Wi-Fi router at home and connect all devices

I could actually understand a lot about hardware components and used this to build my own PC by checking the specifications

I was able to understand a task given in my internship program regarding networking

I learnt about how database is working and applied it for my intern project

I applied my IT infrastructure Management skills by setting up a basic file sharing netv work between computers : easy project collaboration

Fig .12. Students' self-reported example applications of skills learnt from the IT Infrastructure Management course

Even though the study is based one specific course in an institution, the CT-based framework is inherently transferable. The identical principles of mapping can be used in the case of courses that impart computing infrastructure education. Eg., System Administration , Cloud Computing, Network Management, Datacenter Administration etc., where the concepts of Decomposition, Abstraction, and Algorithmic Reasoning become basic building blocks of configuration, automation and architectural decision making.

CONCLUSION

The integration of CT skills into the IT Infrastructure Management course has proven to be both effective and pedagogically sound, as evidenced by the strong alignment between quantitative and perception-based evaluations. The mapping of activities to CT components enabled targeted skill development, while statistical validation ensured the reliability of the assessment framework. High performance averages, tight score distributions, and positive student perceptions collectively indicate that the approach not only fostered deep conceptual understanding but also enhanced students' ability to apply CT in practical scenarios.

The correlation between Pattern Recognition, Abstraction, and Algorithmic Thinking underscores the interdependence of these skills in problem-solving contexts. While most students adapted well, the small percentage of outliers highlights the need for adaptive instructional strategies to support varied learning paces. Future work will explore the scalability of this pedagogical framework across other computing infrastructure courses and investigate the longitudinal impact of CT integration on graduates' professional performance. Further, with multi-institutional studies the generalizability of the proposed framework can be validated.

REFERENCES

Sachs, G. (2024). AI is poised to drive 160% increase in data center power demand. Goldman Sachs, 14.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition.

J. Aldino, et al., “Learner-centered feedback analytics in higher education: A large-scale case study,” *Computers & Education*, vol. 182, p. 104467, 2022.

Y. Wang, et al., “Computational thinking and creativity: A scoping review,” *Computers & Education*, vol. 172, p. 104271, 2021.

C. Lin, et al., “Learner-centred analytics of feedback content: Insights for improving educational interventions,” *British Journal of Educational Technology*, vol. 54, no. 3, pp. 1012–1030, 2023.

R. Johnston, et al., “Learning analytics in computing education: A systematic review of emerging trends and frameworks,” *Journal of Systems and Software*, vol. 209, p. 111919, 2024.

M. Lee and J. Kim, “Designing a self-determination theory-informed learning analytics dashboard to enhance student engagement in asynchronous online courses,” *Journal of Computing in Higher Education*, 2024.

K. W. Huang, et al., “Learning analytics dashboards in health professions education: Usability, satisfaction, and visual design considerations,” *Advances in Health Sciences Education*, 2025.

R. Kaliisa, et al., “Students’ engagement with analytics feedback in higher education: Implications for design and practice,” *International Journal of Educational Technology in Higher Education*, vol. 22, no. 1, pp. 1–24, 2024.

R. Johnston, et al., “Predicting student engagement in computing modules using interpretable machine learning,” *arXiv preprint*, arXiv:2412.11826, 2024.

R. Sadallah, “Adaptive understanding framework: Towards learner-centered learning analytics dashboards,” *arXiv preprint*, arXiv:2505.12064, 2025.

A. Adeyemi and S. AlOtaibi, “Adaptive decision support for real-time student feedback using LightGBM and SHAP explainable AI,” *arXiv preprint*, arXiv:2508.07107, 2025.

J. Zhou, et al., “Tag-based automated feedback generation for students using ChatGPT: A teacher evaluation study,” *arXiv preprint*, arXiv:2501.06819, 2025.

S. Holstein and A. Cohen, “Scratch teachers' perceptions of teaching computational thinking with school subjects in a constructionist approach,” *Thinking Skills and Creativity*, vol. 56, p. 101772, 2025, doi: 10.1016/j.tsc.2025.101772.

C. N. Hirt, T. D. Eberli, J. T. Jud, A. Rosenthal, and Y. Karlen, “One step ahead: Effects of a professional development program on teachers’ professional competencies in self-regulated learning,” *Teaching and Teacher Education*, vol. 159, p. 104977, 2025, doi: 10.1016/j.tate.2025.104977.

Y. Liu, M. A. Llorens, Y. Kong, C. Teoh, and D. J. Barnes, “A systematic review of K-12 teachers’ professional development for teaching computational thinking,” *Disciplinary and Interdisciplinary Science Education Research*, vol. 6, no. 1, p. 27, Jun. 2024, doi: 10.1186/s43031-024-00172-x.

R. Neves Rodrigues, C. Costa, and F. M. L. Martins, “Integration of computational thinking in initial teacher training for primary schools: a systematic review,” *Frontiers in Education*, vol. 9, 2024, doi: 10.3389/feduc.2024.1330065.

L. Greifenstein, U. Heuer, and G. Fraser, “Exploring programming task creation of primary school teachers in training,” *arXiv preprint*, arXiv:2306.13886, 2023.

P. Varela, M. F. Prieto, and A. R. Ariza, “Assessing computational thinking skills in engineering education: A mixed-methods approach,” *Computers & Education*, vol. 191, pp. 104–135, 2023.

F. Ali and J. Smith, “Cross-case analysis of computational thinking integration in K–12 curricula,” *Journal of Educational Computing Research*, vol. 61, no. 1, pp. 72–94, 2023.

P. Shah, R. Thomas, and S. Chan, “A systematic review of computational thinking professional development initiatives,” *Education and Information Technologies*, vol. 29, no. 2, pp. 1125–1150, 2024.

Y. Wu and H. Li, “Computational thinking as a data-centric literacy: Framework and implications,” *Journal of Computer Assisted Learning*, vol. 40, no. 3, pp. 755–772, 2024.

E. Yeni, K. W. Lai, and S. M. Tan, “Interdisciplinary integration of computational thinking in K-12 education: A systematic review,” *Education and Information Technologies*, vol. 29, no. 7, pp. 8357–8381, 2024.

G. Falloon, “Building young children’s computational thinking capability through problem-based learning,” *Computers & Education*, vol. 203, 104898, 2024.

K. Subramaniam, D. Hammer, and L. X. Wang, “STEM ways of thinking: A design-based research study on engineering design-based problem solving in physics,” *arXiv preprint arXiv:2503.05957*, 2025.

S. Adorni, G. M. Rosa, and R. M. Bottino, “FADE-CTP: A framework for the analysis of computational thinking problems in education,” *arXiv preprint arXiv:2403.19475*, 2024.

M. Haugen and T. Stålhane, “Challenges in DevOps instruction: Academic and industry perspectives,” *Proc. ACM/IEEE Software Engineering Education and Training*, 2022.

M. Haugen, T. Stålhane, and M. F. Johansen, “Overcoming DevOps instructional challenges through project-based learning,” *IEEE Trans. Educ.*, vol. 66, no. 4, pp. 543–554, 2023.

Gransbury, I., Brock, J., Root, E., Catete, V., Barnes, T., Grover, S., & Ledeczi, Á. (2023). Project-based software engineering curriculum for secondary students. *Proc. WiPSCE ’23*.

Afshar, Y., Moshirpour, M., Marasco, E., Kawash, J., Behjat, L., & Moussavi, M. (2022). An integrated SE curriculum through PBL. *ASEE Annual Conf. & Exposition*.

Garcés, L., & Oliveira, B. (2024). Teaching SE with PBL: A four-year experience report. *SBES Proceedings*.

Iyer, G. N., Goh, A., Chee, M. H. E., Choong, W., & Koh, S. W. (2024). A web-based IDE for DevOps learning in HE. *TALE 2024*.

Garcia, P. S. C., Ferreira, J., Gonçalves, M., Carneiro, T., Figueiredo, E., & Pereira, I. M. (2024). Current DevOps teaching techniques: A systematic review. *SBES Proceedings*.

Borja-Fernández, G., et al. (2023). Automatic feedback and assessment of team-coding assignments in DevOps context. *Int. J. Educ. Technol. Higher Educ.*, 20, 95–11.

Bonetti, T. P., Silva, W., & Colanzi, T. E. (2025). Example-based learning in software engineering education: A systematic mapping. *arXiv preprint*, arXiv:2503.18080.