A Micro-Curriculum for Training Undergraduate Interdisciplinary Scientists

Javed I. Khan[1] and Philip Thomas[2]
e-mail: Javed@ kent.edu | plthomas@ kent.edu

[1] Internetworking and Media Communications Research Laboratories
Department of Computer Science

[2] Information Systems Division
Kent State University
233 MSB, Kent, OH 44242
March 2025

Abstract

In this document we present the design of a **micro-curriculum** aimed at preparing undergraduate students to work effectively in interdisciplinary, computation-intensive research environments. It is built around three core pillars: **(a)** breaking silos through improved communication across scientific and computational domains fostering collaboration, **(b)** developing an understanding of the rich scientific ecosystem—the shared experiment facilities and labs, software, infrastructures, tools, protocols, and modalities of operation, and **(c)** cultivating indispensable scientific values such as **reproducibility, transparency, ethical responsibility, and sound data management** within this ecosystem. To provide structure and continuity across these themes, we employ **SNOWFLAKE [11,12]**—a formal language for describing and operationalizing scientific workflows that integrate human, computational, machine, and informational processes. Using SNOWFLAKE alongside experiential learning, students learn not only how to participate in scientific projects but also how to **formally represent, document, and reason about their structure, execution, and reproducibility**. Below, we outline the proposed curriculum and its implementation framework. We also test a new teaching method in which students record concepts in a way that both humans and machines can understand, by encoding them into a conversational AI. Their project artifact becomes a universal communicator—an embodiment of *Teaching Science to Describe Itself*.

1. Introduction

The complexity of modern science increasingly demands interdisciplinary fluency — the ability to integrate computational, analytical, and domain-specific reasoning across diverse fields. Yet, undergraduate education remains largely discipline-siloed, producing students who are strong in their majors but often unequipped to collaborate effectively in real-world, cross-domain research teams. A *micro-curriculum*—a compact, modular learning program embedded within or alongside traditional coursework—addresses this gap by providing just-in-time, applied, and cross-cutting competencies that prepare students for collaborative research environments. A micro-curriculum dedicated to interdisciplinary scientific integration provides the missing bridge. It trains CS students not only to *code for computation* but to *think for collaboration*—to understand experimental workflows, data provenance, FAIR principles, and the epistemic purpose behind scientific computing. By learning to engage meaningfully with non-CS researchers, undergraduate computer scientists become architects of discovery, capable of designing reproducible, transparent, and ethically responsible computational systems that advance both science and society.

2. Why is it especially critical for Computer Science Undergraduates?

3. Why current/traditional way of training have gaps?

Traditional scientific education developed over few centuries, was designed for a world where computation was peripheral—a supporting metal tool for analysis rather than an integral part of discovery itself. The skills were passed on by informal means. In that model, computer science, data management, and ethical governance evolved as separate silos, leaving most domain scientists underprepared for the demands of today's hybrid research environment. As a result, researchers often rely on ad hoc scripts, opaque data pipelines, and undocumented workflows that are difficult to reproduce, share, or secure. The gap is not in intelligence or motivation but in structure: students are taught *how to compute*, but rarely *how computation transforms the scientific process*.

This separation has fleetingly created few critical vulnerabilities in contemporary science exploration. Unlike classical experiment, experiments are often non-reproducible because computational steps are not formally documented; sensitive data are mishandled because privacy and cybersecurity are treated as peripheral concerns; and interdisciplinary projects stall because collaborators lack a shared language for representing scientific intent, dependencies, and accountability. In effect, science has become computationally powerful but epistemically fragile—able to produce results quickly but not always to explain or reproduce them.

Below, we outline the proposed curriculum.

4. Approach

While designing this micro-curriculum, we adopt a new approach—*Teaching Science to Describe Itself.* This marks a pedagogical shift from merely training students to use computational tools toward cultivating their ability to articulate, structure, and reason about the scientific process itself. Students engage with live scientific projects, learning not only to interpret and express their own understanding, but also to render the learned concepts and essential fine ideas intelligible both to other humans and to machines.

5. Learning Objective

1. Communicate effectively with interdisciplinary researchers to understand scientific goals and computational challenges.
2. Translate scientific problems and methods into structured computational workflows.
3. Operate Linux and HPC systems, including file management, permissions, and job scheduling.
4. Install, configure, and optimize scientific software with appropriate dependency and version management.
5. Use major scientific packages (e.g., Python, R, MATLAB, GROMACS, COMSOL) for domain-specific analysis.
6. Design and automate computational workflows using scripting and workflow managers such as Slurm, PBS, or Nextflow.
7. Evaluate workflow scalability, parallelization, and containerization for HPC or cloud environments.
8. Describe and utilize institutional and national computing infrastructures for research execution.
9. Apply best practices in data management, security, and version control to ensure reproducibility and transparency.
10. Demonstrate ethical and responsible research conduct, including FAIR data principles and IRB awareness.
11. Integrate all technical, scientific, and ethical components into a comprehensive workflow representation and final portfolio.

6. Module Descriptions

Module / Day	Theme	Core Topics
Module 1 – Communication, Research Context, and Workflow Languages	Understanding scientific teams, collaboration, research problems, and translating them into computational workflows	• Interdisciplinary collaboration and communication skills • Active listening and requirement gathering • Describing scientific problems in computational terms • Introduction to workflow languages (CWL [1], WDL [2], Nextflow[3]) • Role of the "digital scientist"
Module 2 – Linux & HPC Fundamentals + Facility Tour	Orientation to the HPC environment and physical computing infrastructure	• Linux shell, directory structure, and permissions • Cluster architecture: login, compute, and storage nodes • Job schedulers (Slurm[4], PBS [5]) • Facility overview (data centers, interconnects, cooling, GPUs) • Environment modules and resource access
Module 3 – Installing & Optimizing Scientific Software	Software environments, compilers, and performance tuning	• Software modules, Conda/virtual environments • Compilers and build systems • Installation and configuration of scientific tools • Dependency management • Performance profiling and optimization
Module 4 – Scientific Packages	Common research software stacks and scripting for computation	• Python, R, and MATLAB for scientific data processing • Input/output architecture of domain packages • Automation with shell scripts and batch files • Logging and error handling
Module 5 – Scientific Computing Infrastructure	National and institutional computing ecosystems	• Overview of supercomputers and national labs (ACCESS[8], OSG [5], DOE, NSF) • Campus HPC systems & shared resources • Accounts, allocations, and policies • Data transfer tools (Globus[6], rsync) • Support networks
Module 6 – Automating Scientific Computing	Workflow design and automation in computational research	• Workflow concepts and design patterns • Automation with Python/R/MATLAB • Batch and pipeline scripting (Slurm, PBS) • Restart and error-handling strategies • Workflow managers (Snakemake, Nextflow)

Module 7 – Transitioning to Scalable Computing	Scaling and integration of workflows into HPC and cloud environments	• Migrating from desktop to HPC/cloud • Parallel computing basics (MPI, OpenMP) • Container technologies (Docker, Singularity) • Hybrid/distributed models • Project handoff and reflection
Module 8 – Data Management and Data Security	Organizing, versioning, and publishing scientific data and workflows	• Sensitive data handling and security • Data organization and naming conventions • Version control (Git/GitHub) • Metadata and logging for reproducibility • Data publishing and IP awareness
Module 9 – Responsible Computing – Ethics, IRB, Reproducibility & Open Science	Responsible and ethical computing in collaborative research environments	• Research ethics and responsible conduct • Institutional Review Board (IRB) principles • Reproducibility and F-A-I-R framework [9] • Open Science ecosystems and data sharing [10] • Attribution and research integrity (ESPS 2023)
Module 10 – Capstone Presentations & Reflection	Integration of scientific, computational, and ethical learning	• Capstone presentation of project documentation • Peer review and feedback session • Reflection on computational practice and teamwork • Final submission of intern portfolio

7. SNOWFLAKE Exercise

To provide structure and continuity across these themes, one can employ something like **SNOWFLAKE** [11,12]—a formal language for describing and operationalizing scientific workflows that integrates human, computational, machine, and informational processes. Using SNOWFLAKE alongside experiential learning, students learn not only how to participate in scientific projects but also how to **formally represent, document, and reason about their structure, execution, and reproducibility**.

At the top level SNOWFLACK is a workflow description language- but it contains a extensive and rich set of semantic information normally associated with a typical science workflow, including human roles to cognitive information objects- not addressed in other Workflow languages. The semantic richness of SNOWFLAKE arises from the diversity and granularity of information it encapsulates across all scientific process dimensions. Each element carries not just operational metadata—such as execution parameters, resources, or timing—but also contextual, relational, and ethical descriptors that articulate *why* an action occurs, *who* is responsible, *under what conditions*, and *with what standards of integrity and reproducibility*.

The schema unites technical, cognitive, and institutional semantics—linking scientific goals, human decisions, computational steps, and data transformations within a single interoperable framework. Attributes such as accountability entities, verification methods, FAIR indices, and ethical compliance markers enrich the record beyond mechanical provenance, transforming SNOWFLAKE into a living semantic fabric that reflects the full epistemic and social context of science in action.

Below, we outline the proposed curriculum and its implementation framework. It is helpful to use **SNOWFLAKE** and a real scientific project as a cognitive exercise for developing structured thinking. We decomposed SNOWFLAKE into several incremental profiles, and students can be guided to progressively build and populate these profiles using provided worksheets.

Module	SNOWFLAKE Integration Exercise
Module 1 – Communication, Research Context, and Workflow Languages	Extract **SNOW–(H + S)** TEAM, COLLABORATION, and SCIENCE elements. Complete the initial workflow skeleton with scientific context and collaboration structure.
Module 2 – Linux & HPC Fundamentals + Facility Tour	Map **SNOW–(M)** INFRASTRUCTURE context to the facility. Capture architectural and resource layers encountered during the campus or national HPC tour.
Module 3 – Installing & Optimizing Scientific Software	Extract **SNOW–(C + I)** SOFTWARE and COMPUTATION elements; refine element-level narratives describing computational dependencies and software configuration.
Module 4 – Scientific Packages	Populate **SNOW–(C + I)** RESOURCE parameters corresponding to application architecture, runtime dependencies, and execution environment.
Module 5 – Scientific Computing Infrastructure	Populate **SNOW–(M)** INFRASTRUCTURE mappings to document workflow execution contexts and hardware configurations.
Module 6 – Automating Scientific Computing	Extract **SNOW–(E + M)** ENGINEERING and cluster provisioning parameters; represent job scheduling, automation, and containerization attributes.
Module 7 – Transitioning to Scalable Computing	Recast the workflow into a **SNOW–HPC** configuration for scalability evaluation and performance mapping.

Module 8 – Data Management and Data Security	Populate **SNOW–(I + S)** DATA MANAGEMENT and SECURITY attributes; evaluate data integrity, privacy, and compliance constraints.
Module 9 – Responsible Computing – Ethics, Reproducibility & Open Science	Populate **SNOW–(R)** RESPONSIBLE SCIENCE attributes addressing reproducibility, FAIR compliance, IRB classification, and ethical disclosure.
Module 10 – Capstone Presentations & Reflection	Final Capstone Deliverable — Present a comprehensive **SNOWFLAKE Project Narrative** summarizing the scientific, computational, and ethical dimensions. Submit final portfolio.

8. Evaluation & *Teaching Science to Describe Itself*

The structured SNOWFLAKE project model can be further entered into a modern conversational AI system, allowing an evaluator to query the AI model about the project in natural language to assess the internalization of the project- reflecting on student's own understanding and communication. This approach represents a pedagogical shift—from merely training students to use computational tools toward cultivating their ability to articulate, structure, and reason about the scientific process itself. Through live scientific projects, students learn not only to interpret and express their understanding, but also to encode their insights so they are intelligible both to other humans and to machines. When stored in this form, the student artifact itself becomes a *project communicator*—an embodiment of the principle of **Teaching Science to Describe Itself**.

9. Conclusions

The curriculum has been exercised and refined over the past three years at Kent State University. A select group of Choose Ohio First–Computer Science (COF-CS) undergraduate research scholars were embedded within campus research teams utilizing the NSF-funded ICE Cluster. These student researchers conducted the formalized narration of active scientific projects using early versions of the SNOWFLAKE schema, documenting the structure, intent, and interconnections of complex research workflows. Their reflections and systematic feedback were instrumental in shaping successive iterations of the framework.

At institutions like Kent State University, where undergraduate researchers are directly engaged in computationally intensive, data-driven projects, a micro-curriculum has offered several advantages:

1. **Rapid Onboarding to Research Practice:** It early introduced freshman undergraduate students to real scientific workflows, data management, ethical considerations, and computational reproducibility—skills typically learned only through graduate training or professional experience.

2. **Bridging Disciplinary Languages:** Students learned to translate between computational logic, experimental design, and scientific communication, making them more effective collaborators in mixed teams (e.g., computer science, biology, and psychology)- using the common language of science - *critical thinking*.

3. **Applied Learning through Research Integration:** The micro-curriculum integrated academic instruction with live project participation, converting learning into immediately applicable research skills.

In short, the micro-curriculum serves as an **educational bridge** between classroom learning and scientific collaboration. It transforms undergraduates from passive learners into **active contributors** within interdisciplinary research ecosystems, cultivating a new generation of scientists fluent in both disciplinary depth and cross-domain integration.

10. Acknowledgements

We gratefully acknowledge our undergraduate researchers — Alexis Faudree, Brandon Renner, Avi Rathod, Andrew Lindsey, Annika Hall, Ben Leber, Brandon Valleau, Jason Graham, Makayla Henninger, Michael Moses, and Samuel Ruby — for their dedication and insight. We also extend our sincere thanks to the project directors and research teams who generously allowed the students to shadow and collaborate in their work while developing experimental SNOWFLAKE workflows, including Dr. Sangeet Lamichaney, Dr. Hamza Balci, Dr. Kuldeep Singh, Dr. Angela Ridgel, Dr. Joseph Ortiz, Dr. Jong-Hoon Kim, Dr. Xiang Lian, Dr. Robert Clements, Dr. Michael Strickland, and Dr. Enrico Gandolfi. We especially thank Ms. Jeanne Tan for coordinating the many logistical and operational aspects of this multidisciplinary collaboration. These student artifacts were presented in Choose Ohio First Poster Conference 2023, 2024 and 2025.

This project has been supported by the National Science Foundation (NSF Award #2201558 and NSF Award #1925678), with additional engineering contributions from the Division of Information Technology and facilities support from the Department of Computer Science. The State of Ohio Department of Education's Choose Ohio First–Computer Science (COF-CS) program provided supported the participating student researchers. The ICE Cluster has also received supplementary funding from the Division of Research and Economic Development.

11. References

[1] Common Workflow Language (CWL): Amstutz, P., Crusoe, M.R., Tijanić, N., Chapman, B., Chilton, J., Heuer, M., Soiland-Reyes, S. & Khan, F., 2016. *Common Workflow Language, v1.0.* Figshare. DOI: 10.6084/m9.figshare.3115156

[2] Workflow Description Language (WDL) / Cromwell: Voss, K., Gentry, J., Van Der Auwera, G. & O'Connor, B., 2017. The Workflow Description Language (WDL) and Cromwell Workflow Engine. Broad Institute. Available at: https://openwdl.org

[3] Nextflow: Di Tommaso, P., Chatzou, M., Floden, E., Barja, P.P., Palumbo, E. & Notredame, C., 2017. *Nextflow enables reproducible computational workflows. Nature Biotechnology,* 35(4), pp. 316–319. DOI: 10.1038/nbt.3820

[4] Slurm Workload Manager: Jette, M.A., Yoo, A.B. & Grondona, M., 2002. *SLURM: Simple Linux Utility for Resource Management. Lecture Notes in Computer Science (Job Scheduling Strategies for Parallel Processing),* 2862, pp. 44–60. DOI: 10.1007/10968987_3

[5] Portable Batch System (PBS): Henderson, R., 1995. *Job scheduling under the Portable Batch System. Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing,* Springer, pp. 279–294. DOI: 10.1007/BFb0026619

[5] Open Science Grid (OSG): Pordes, R., Petravick, D., Kramer, B., Oliveira, R., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Wright, D. et al., 2007. *The Open Science Grid. Journal of Physics: Conference Series,* 78, 012057. DOI: 10.1088/1742-6596/78/1/012057

[6] Globus Data Transfer: Foster, I., Allen, G., Kesselman, C., Lederer, H. & Nick, J., 1999. *The Globus Toolkit for High Performance Distributed Computing. Concurrency and Computation: Practice and Experience,* 11(13), pp. 109–128. DOI: 10.1002/(SICI)1096-9128(199911)11:13<109–128::AID-CPE553>3.0.CO;2-Q

[7] Docker: Merkel, D., 2014. *Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal,* 2014(239), pp. 2. Available at: https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment

[8] ACCESS (Advanced Cyberinfrastructure Coordination Ecosystem Services and Support): Townsend, R.M., Foster, I.T. & Stewart, C.A., 2022. *ACCESS: A new model for U.S. national cyberinfrastructure. Computing in Science & Engineering,* 24(3), pp. 94–102. DOI: 10.1109/MCSE.2022.3150878

[9] FAIR Computational Workflows: Goble, C.A., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M.R. & Peters, K., 2020. *FAIR computational workflows. Data Intelligence,* 2(1-2), pp. 108–121. DOI: 10.1162/dint_a_00033

[10] RO-Crate: Soiland-Reyes, S., Bacall, F., Owen, S., Lusher, S. & Goble, C.A., 2022. *Packaging research artefacts with RO-Crate. Data Science Journal,* 21(1), p. 10. DOI: 10.5334/dsj-2022-010

[11] Javed I. Khan and Philip Thomas, (2025), *An Exploration into Scientific Narrative Of WorkFLow for Learning, Analytics and Knowledge Engineering*, Technical Report 2025-03-01 Internetworking and Media Communications Research Laboratories, Department of Computer Science, Kent State University, available from: http://medianet.kent.edu/technicalreports.html

[12] Javed I. Khan and Philip Thomas, (2025), *Semantic Container for Scientific Narrative Of WorkFLow for Learning*, Analytics and Knowledge Engineering, Technical Report 2025-03-02 Internetworking and Media Communications Research Laboratories, Department of Computer Science, Kent State University, available from: http://medianet.kent.edu/technicalreports.html

[13] Javed I. Khan and Philip Thomas, (2025), *A Micro-Curriculum for Training Undergraduate Interdisciplinary Scientists*, Technical Report 2025-08-02 Internetworking and Media Communications Research Laboratories, Department of Computer Science, Kent State University, available from: http://medianet.kent.edu/technicalreports.html