



doi:10.5281/zenodo.18754652

Design of a Structured-Iterative Hybrid Model for Enhancing Java Programming Skills among Undergraduate Students

Oruan Memoye Kepeghom PhD.

Department of Computer and Robotics Education.
Federal College of Education Technical Omoku, Rivers State, Nigeria

¹oruanmem@gmail.com, ²memoye.oruan@fctomoku.edu.ng

ORCID: 0000-0001-5746-1869

ABSTRACT

Teaching programming pedagogy efficiently in computer education especially the conceptual understanding and practical skills mastery remains a core and major challenge coupled with poor infrastructure in most of the higher institutions. This study critically proposes a Structured-Iterative Hybrid Model (SIHM) fuses the systematic Top-down sequencing of the Waterfall model with the adaptive, bottom-up feedback-driven cycle of the Agile methodology to enhance Java programming core concepts instruction at the undergraduate level. Taking insight on recent research in hybrid instructional models and software development pedagogy, the design for this paper will details on the conceptual framework, implementation strategy, expected outcomes, and implications for curriculum innovation. The hybrid model SIHM target is to improve students' understanding, comprehension of concepts, iterative programming and coding ability, problem—solving skills, and readiness for collaborative software development tasks.

Keywords: Hybrid Model, Agile, Waterfall, Java Programming, Undergraduate Education, Instructional Design, Software Engineering Pedagogy.

1. INTRODUCTION

Programming and the implementation of it which is coding remains the driving force in software development. Java is one of the widely used programming languages taught in tertiary computing curricula due to its rich object-oriented foundations, extensive libraries, and industrial relevance.

In the same thought Aljahdali et al., 2017 agreed that Java remains foundational due to its widespread industry adoption, object-oriented paradigms, and relevance in software engineering courses. Notably, studies has shown that persistent challenges in effectively teaching and learning Java, include high cognitive load, students' difficulty in transferring theoretical concepts to practical coding task, and a shallow engagement with iterative problem-solving practices (Lahtinen, Ala-Mutka & Jarvinen, 2005; Rountree & Rountree, 2003)

Notwithstanding of it rich functionality undergraduate students often struggle with drafting from conceptual or theoretical understanding to practical code implementation, design and project development (Spurrier & Topi, 2023). Basically the Waterfall model align to the linear instructional models, sequential topics rigidity and fail to support repeated practice and refinement as posit by (Mishra & Alzoubi, 2023). Alternatively, the pedagogy as practiced by the Agile model methodology emphasize iterative feedback

and adaptability, which motivate well with programming practice but may lack coherent coverage of functionality concepts as emphasized by the Waterfall-based model.

The underlining facts brings the hybrid instructional models that blend structured content delivery with iterative practice cycles have emerged in education to checkmate and balance consistency and responsiveness (Astawa & Purwaningrat, 2025). It will be an effective blend in software engineering education to develop a hybrid process models for instance the merging of the Waterfall and Agile model to yield a fruitful results in teaching and learning (Yahya & Maidin, 2023; Maharao, 2022).

The structured emphasis of the SIHM states that learning objectives, modular content delivery and systematic progression from foundational to advanced topics. This approach draws from the concept or principles of Instructional System Design (ISD), it ensures that programming concepts such as control structures, object-oriented design, exceptional handling, and data structure are introduced in a logical, building-block fashion (Gagne et al 2005)

An overview, the structured-iterative Hybrid Model inculcate principles from

- Instructional Design
- Learning Sciences and
- Software education;

To create a powerful and dynamic framework for enhancing Java programming skills among undergraduates. The core emphasis on scaffold content delivery and iterative mastery offers a research-informed pathway to addressing longstanding pedagogical challenges in programming education.

2. LITERATURE REVIEW

2.1 Hybrid Models in Education

Hybrid models are strong and better systems compared with standalone models. Hybrid instructional frameworks that combine multiple educational approaches have gained grip in enhancing student engagement and learning outcomes. Such frameworks actively integrate face-to-face, online, and iterative learning activities (for example, blended/hybrid formats) that support varied learner needs (Manyasi, 2025;Roig et al., 2025). This paper work focuses on delivery formats for both online and offline ie in-person learning. The bottom-line concept is combining structures to maximize learning outcome in view of the rationale for hybrid instructional design model

2.2 Software Development Hybrid Models

In software engineering research, hybrid methodologies combine structured, plan-driven models like Waterfall with iterative frameworks such as Scrum or Agile to leverage the planning benefits of one and the flexibility of the other model (Yahya & Maidin, 2023; Maidin & Yahya, 2023). These hybrid models are often considered in situations where project requirements demand both upfront planning and iterative adaptation.

2.3 Pedagogical Challenges in Programming Education

Finding has shown some challenges in programming pedagogy especially in teaching and learning, ranging from understanding abstraction (e.g. objects, recursion, and polymorphism), translating problem statements into algorithms and debugging logical errors. More studies reveals that the challenges spans from students' anxiety with abstract concepts, lacking of debugging skills, and less progress from small code examples to complex larger industrial projects. Agile model with emphasis on iterative practices, frequent feedback, and learning collaboration are identified as major pedagogical strategies (Spurrier & Topi, 2023).

3. RESEARCH METHODOLOGY

Design and development research (DDR) methodology was used to conceptualize the SIHM and showcase it major components. The setup of the model's creation draws insights from hybrid methodology literature, instructional design principles, and software development pedagogy. The resources for the methodology of this study spans from peer-reviewed literature, Java programming core concepts literatures, software engineering education source research work and hybrid learning research

work. The purposed was to build a basic and core knowledge required to for hybrid instructional architectural framework for Java programming.

4. Design of the Structured-Iterative Hybrid Model (SIHM)

The hybrid model consist of three main phases:

4.1 Phase 1: Structured Curriculum Sequencing

This phase is based on modeling on plan-driven methodology, emphasis on logical sequence lesson delivery of core Java concepts and topics. Inclusive are object-oriented concept such as classes, object, inheritance, polymorphism, exception handling etc. This is the basic conceptual building stage that involves sequential conceptual assimilation before engaging in iterative practice.

4.2Phase 2: Iterative Coding Sprints

Using Agile arrangement of iterative cycles, students work in groups in short sprints that are as follows;

- Mini-projects based on recently taught concepts
- Daily or weekly progress checks
- Pair programming and peer review
- Instructor feedback loops

Each sprint may last for 1-2 weeks and progress with time

4.3 Phase 3: Integrated Project Delivery

Students apply cumulative learning in a capstone project (e.g., a Java-based application involving GUI, database integration, and modular design). Work is completed incrementally, with each sprint increasing the project's functionality.

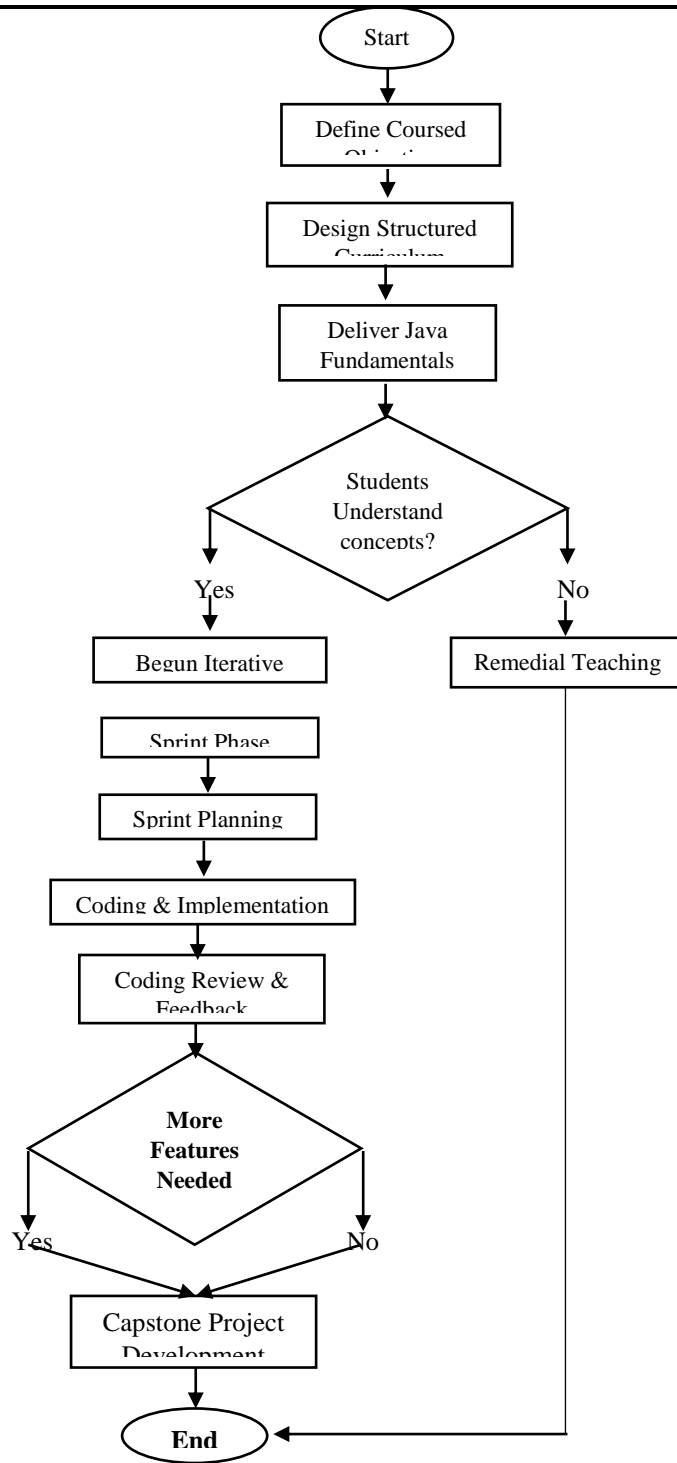


Figure 1: Overall Hybrid

5. Expected Benefits

The SIHM is expected to:

- No doubt the structural delivery pedagogy will improve programming concepts retention.
- It will enhance practical coding skills via iterative project development.
- Fortify problem-solving and debugging ability through continuous feedback.
- Boost collaboration and communication skills via pair programming and peer review. These results are reliable with the literature emphasizing active learning and iterative practice in computing education.

6. DISCUSSION

The collaboration of the structured sequencing and iterative practice solves several pedagogical gaps in traditional programming instruction delivery. In like manner as hybrid software development models blend governance with flexibility to achieve better project outcomes (Mishra & Alzoubi, 2023), SIHM balances consistency, reliability and adaptability to support student learning trajectories.

7. CONCLUSION

The Structured–Iterative Hybrid Model (SIHM) offers a pragmatic instructional design for enhancing Java programming skills among undergraduates. Its amalgam of structured content progression with iterative, feedback-driven practice aligns with both educational theory and software engineering practice. This model is an intervention and provides a promising improvement and a way forward for curriculum innovation that bridges the gap between theoretical understanding and practical proficiency.

REFERENCES

1. Aljahdali, H., Almazroi, A. A., & Alshammari, R (2017). Empirical evaluation of Java as first programming language. *International Journal of Computer Applications*.
2. Astawa, I. P. P., & Purwaningrat, P. A. (2025). The Future of Learning: Exploring Hybrid Educational Models and Their Impact on Student Engagement and Performance in a Digitalized World. *International Journal of Research and Innovation in Applied Science*.
3. Gagne, R.M., Wager, W.W., Golas, K.C & Keller, J.M (2005). *Principles of instructional design*. Wadsworth/Thomson Learning.
4. Lahtinen, E., Ala-Mutka, K., & Jarvinen, H.M (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14-18.
5. Maidin, S. S., & Yahya, N. (2023). An Exploration into Hybrid Agile Development Approach. *International Journal of Advanced Science and Computer Applications*.
6. Maharao, C. S. (2022). A Comparative Analysis of Agile, Waterfall, and Hybrid Methodologies in Software Project Success. *ShodhKosh: Journal of Visual and Performing Arts*.
7. Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: a comparative analysis. *International Journal of System Assurance Engineering and Management*.
8. Manyasi, B. N. (2025). Blended Learning; Models for Transforming Instructional Practice. *International Journal on Integrating Technology in Education*.
9. Robins, A., Rountree, J., Rountree, N (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137-172.
10. Roig, P. J., et al. (2025). Study on Hybrid Education in Terms of Space, Time, Language, and Frameset. *Education Sciences*.
11. Spurrier, G., & Topi, H. (2023). Teaching How to Select an Optimal Agile, Plan-Driven, or Hybrid Software Development Approach: Lessons from Enterprise Software Development Leaders. *Journal of Information Systems Education*.