Preliminary Design Review

MISCE project

Mechatronics for Improving and Standardizing Competences in Engineering

Competence: Computer Programming

Workgroup: University de Castilla-La Mancha

University of Žilina

This document is the Preliminary Design Review of the technical competence ‘Computer Programming’. Its briefly describes the set of programming exercises proposed within the MISCE project, intended to be implemented across several programming environments to support and enhance the acquisition of this competence in engineering degree programs.

Version: 1.0

Date: September 2th, 2024

Visit https://misceproject.eu/ for more information.

Index of contents

Index of figures

-

Index of tables

1 Competence and skills

The conceptual design presented in this document is referred to the technical competence:

C1. Computer Programming

which related skills are (see Table I):

Table I. Skills of Computer Programming

S1.1.	To be able to use proper data structures
S1.2.	To be able to develop programs to solve engineering problems
S1.3.	To be able to characterize the performance of programs
S1.4.	To be able to debug faulty programs
S1.5.	To be able to optimize programs

The different types of programming exercises and environments presented in this document have been analysed to ensure that they can effectively support the development and improvement of the aforementioned competence.

2 Proposal

For this competence, MISCE project proposes different types of programming exercise to be developed in several programming environments.

It is well known that the execution of diverse programming exercises significantly enhances students' algorithmic thinking, computational problem-solving, and debugging skills, which are core to the 'Computer Programming' competence. Hands-on tasks—ranging from simple logic puzzles to real-world coding problems—facilitate active learning and deliberate practice, enabling learners to break complex challenges into smaller, solvable steps and receive immediate feedback through code execution. Research indicates that structured problem-solving activities lead to measurable gains in student performance, confidence, and cognitive skills.

By using various programming environments and covering a range of difficulty levels and problem types, the exercises ensure a comprehensive and transferable skillset.

2.1 Types of programming environments

Engineering degrees require programming environments that support both, the development of computational thinking and the practical application of coding skills to solve engineering problems. The selection of these environments is supported by their educational value, applicability to the industry and alignment with specific technical needs of each engineering discipline. The following programming environments or languages constitute the most used in engineering education because of the reasons provide below:

Python	Widely used in engineering education due to its simple, easy-to-learn syntax and extensive collection of scientific libraries. Its ease of use makes it particularly interesting for introducing programming fundamentals to first-year engineering students. Apart form the basics, Python is a powerful tool for numerical analysis (e.g. NumPy, SciPy), data visualization (e.g. Matplotlib), and even machine learning (e.g. TensorFlow, scikit-learn), making it a multidisciplinary programming language
MATLAB/Octave	Heavily used in engineering for matrix-oriented numerical computation, which constitute the foundation of essential topics like control systems, signal processing and system dynamics. Its user-friendly interface and domain-specific toolboxes enable students to solve engineering problems without deal with the complexity associated to lower-level programming languages. Octave is the open-source counterpart, it serves as cost-effective alternative.
C++	Fundamental in engineering education because of its efficiency, hardware-level control and performance. This language is essential in embedded systems, robotics and firmware development, where resource constraints and timing are critical. Learning C++ helps students to understand low-level programming concepts and real-time system design. Additionally, it introduces object-oriented programming preparing students for applications such as simulation engines pr control firmware.
Java	Often included in engineering degrees, particularly in those focused-on software engineering. Its object-oriented design helps students write clear and organize code, and because it can run on different systems without changes, it is a good choice for building programs across platforms. Java is typically used for developing engineering tools with graphical interfaces, simulation environments and backend systems rather than hardware-level programming.

2.2 Types of programming exercises

Programming exercises are a fundamental component of engineering education as they help students develop essential skills such as computational thinking, algorithm design, logical reasoning and problem solving. Through gradually more challenging tasks, students develop the ability to break down complex problems, organise code logically and independently troubleshoot and correct errors.

Incorporating diverse types of exercises into the curriculum helps students go beyond basic understanding, allowing them to absorb key concepts and apply them effectively in practical situations.

In this sense, MISCE project propose the following types of exercises to be execute in different programming environments:

- Preliminary exercises: Matrix/vector manipulation problems.
- Basic Matrix Manipulation Problems.
- Input Argument and Error Checking Problems
- Vector and Matrix Algorithm Problems
- Random Number Algorithm Problems
- String Manipulation Problems
- File Handling Algorithm Problems
- Advanced Vector and Matrix Algorithm Problems

Note that the exercises are arranged in increasing complexity and cover all the skills listed in Table I for the ‘Computer Programming’ competence.

3 Competence and skills analyses

This section has been omitted for the ‘Computer Programming’ competence, as numerous studies and educational practices have shown that the systematic execution of well-designed programming exercises is highly effective in developing the required skills. These exercises inherently promote computational thinking, problem-solving, and code literacy, making additional skill-mapping or analytical breakdowns unnecessary for ensuring meaningful learning outcomes in this area.

References

[1] Qi, H., Phan, A., Liu, H., Lubarda, M., Ghazinejad, M., Gedney, X., ... & Chen, H. (2022, October). Improving engineering students' problem-solving skills through think-aloud exercises. In *2022 IEEE Frontiers in Education Conference (FIE)* (pp. 1-6).

[2] Türker, P. M., & Pala, F. K. (2020). The effect of algorithm education on students' computer programming self-efficacy perceptions and computational thinking skills. International Journal of Computer Science Education in Schools, 3(3), 69–89.

[3] Liu, W., Sun, Y., & Xiong, J. (2024). Modeling students' algorithmic thinking growth trajectories in different programming environments. Smart Learning Environments, 11, Article 24.

[4] Jones, D. J., Madison, K. W., & Wieman, C. E. (2015). Transforming a fourth year modern optics course using a deliberate practice framework. Physical Review Special Topics—Physics Education Research, 11(2), 020108.

[5] Pirzado, F. A., Ahmed, A., Hussain, S., Ibarra-Vázquez, G., & Terashima-Marin, H. (2025). Assessing Computational Thinking in Engineering and Computer Science Students: A Multi-Method Approach. Education Sciences, 15(3), 344. https://doi.org/10.3390/educsci15030344.

[6] Khandait, K. R. (2025). From theory to practice: Integrating problem-solving exercises to strengthen computational thinking. *Journal of Information Systems Engineering & Management, 10*(25s). https://doi.org/10.52783/jisem.v10i25s.3934.