Journal of Information Systems and Technology Management – Jistem USP
Vol. 22, 2025, e202522005
ISSN online: 1807-1775
DOI: 10.4301/S1807-1775202522005 Original Article

A COURSE PLAN USING ACTIVE TEACHING METHODS IN SOFTWARE ANALYSIS AND DESIGN IN THE COMPUTER SCIENCE COURSE

Vitor de Souza Castro[1,2] **,** https://orcid.org/0000-0003-3209-3806
Sandro Ronaldo Bezerra Oliveira[1]**,** https://orcid.org/0000-0002-8929-5145

[1]Universidade Federal do Pará, Belém, PA, Brazil

[2]Universidade Federal do Sul e Sudeste do Pará, Marabá, PA, Brazil

ABSTRACT

The teaching activity requires organization and planning of actions that will be applied in the classroom. Instruments that help the teacher in this process are extremely important to enhance the teaching-learning process. In this sense, the objective of this work is to present a course plan and teaching plan for the discipline of Software Analysis and Design with emphasis on active teaching methodologies. To meet this objective, the following steps were used as a methodology: literature review on the teaching approach of Software Analysis and Design; mapping of competencies between the ACM/IEEE and SBC curricula for the Computer Science course related to Software Analysis and Design; analysis of the main Pedagogical Projects of the IFES Course in Brazil for the Computer Science course; structuring of the course plan and teaching plan for the teaching of Software Analysis and Design; and evaluation based on Peer Review of the two instruments, the course plan and the teaching plan. As main results, the following can be highlighted: the development of the course plan contemplating the main active methodologies identified in the literature, with emphasis on the aspects of Quality in Software Analysis and Design; indication of basic literature for the teaching of Software Analysis and Projects; presentation of methodological procedures; and development of the teaching plan using iconography and resources to stimulate and facilitate students' understanding.

Keywords: Syllabus, Teaching Plan, Software Analysis and Design, Active Methods.

Manuscript first received: 2024-11-30. Manuscript accepted: 2025-09-20

Address for correspondence:
Vitor de Souza Castro. Universidade Federal do Pará, Belém, PA, Brazil; Universidade Federal do Sul e Sudeste do Pará, Marabá, PA, Brazil. E-mail: vitor@unifesspa.edu.br
Sandro Ronaldo Bezerra Oliveira. Universidade Federal do Pará, Belém, PA, Brazil. E-mail: srbo@ufpa.br

INTRODUCTION

For effective teaching, it is crucial that teachers get involved in the planning of activities to be applied in the classroom (Padilha, 2002). The use of instruments that support teachers in this process plays a fundamental role in improving the teaching-learning process. In this scenario, the course plan or curriculum is presented as a central instrument for the teacher to have the direction for the elaboration of the teaching plan.

According to Sacristán (2013), the curriculum is an organized selection of contents to be learned, which, in turn, will regulate the didactic practice that is developed during schooling. Lopes (2014) refers the curriculum to the idea of organization, prior or not, of learning experiences/situations carried out by teachers/education networks in order to conduct the educational process. Therefore, the definition of a curriculum systematically directs the contents and strategies aiming at experiences that strengthen student learning.

The teaching plan is presented as an instance of the course plan (curriculum), containing relevant information about the course schedule, scores of evaluation, methodologies, form of management and monitoring of activities (Gil, 2000). According to Flauzino *et al*. (2021), the construction of a good teaching plan favors a more active, reflective, and critical action of the teacher on his practice and, consequently, instigates him to know more deeply the contemporary educational needs for the training of the professional.

In this context, the construction of the course plan and subsequent the teaching plan strengthen the organization not only of the contents to be treated, but also the methodological approaches to be used and the evaluation strategies.

For the Computer Science course, the understanding of the process of building software is required for the formation of the graduate according to the curricular guidelines (MEC, 2016). Therefore, the inclusion of disciplines that present the contents of this process, considering the generic process defined by Pressman and Maxim (2016) contemplating Communication, Planning, Modeling, Construction and Delivery, becomes fundamental for the training of the graduate.

The modeling stage includes all aspects related to the design of a software. According to Bourque and Fairley (2014), the *Software Design* (SD) area is responsible for the process of defining the architecture, components, interfaces, and other characteristics of a system or component. In addition, this area is an important theme in the training of Computer Science graduates (Garousi *et al*., 2019; Rodríguez-Pérez *et al*., 2021; Leite *et al*., 2020; Aniche *et al*., 2019).

Forti, Breitenbücher and Soldani (2022) present that one of the challenges in the area of Software Engineering is the high heterogeneity of solutions, leading to hybrid systems, with multi-paradigms, and this heterogeneity reflects on how the professional designs the solution, which is directly associated with training in SD.

In the work of Ferreira *et al*. (2018) emerging topics in software engineering were identified and among the most cited are: reuse, software architecture, repositories and *domain-driven-design.* The subjects mentioned are directly related to the study of Software Analysis and Design (SAD), which corroborates the motivation in the development of this work. In addition, the same work by Ferreira *et al*. (2018) indicates that active methodologies are related to alternatives to mitigate students' difficulty related to the contents taught in Software Engineering.

In this sense, the objective of this work is to present a course plan and the teaching plan for the teaching of SAD with an emphasis on active teaching methodologies. To achieve this objective, a literature review was carried out on the teaching approach in *Software Design*, a mapping of competencies between curricula of the *Association for Computing Machinery/Institute of Electrical and Electronic Engineers* (ACM/IEEE) and the Brazilian Computer Society (SBC), an analysis of the main Course Pedagogical Projects of the Federal Institutions of Higher Education (IFES) in Brazil, a structuring of the course plan and teaching plan taking into account the results of the previous stages and an evaluation through peer review.

In addition to this introductory section, this article is organized as follows: Section 2 presents the details of the methodology used; Section 3 presents the proposed course plan for Software Analysis and Design; in Section 4 the teaching plan is presented; in Section 5 the evaluation process of the instruments is presented; in Section 6 the discussions of the results are presented; in Section 7 some related works are presented; and, finally, Section 8 presents the final considerations, limitations and future work.

It should be noted that a previous version of this article was presented at the 20th CONTECSI (Castro and Oliveira, 2024) and the current work is presented as an extension of the study.

METHODOLOGY

This Section aims to present the methodological aspects for the development and evaluation of the course plan and teaching plan for the teaching of SAD. To this end, it was necessary to carry out the methodological steps, as shown in Figure 1, used in the development of the research.

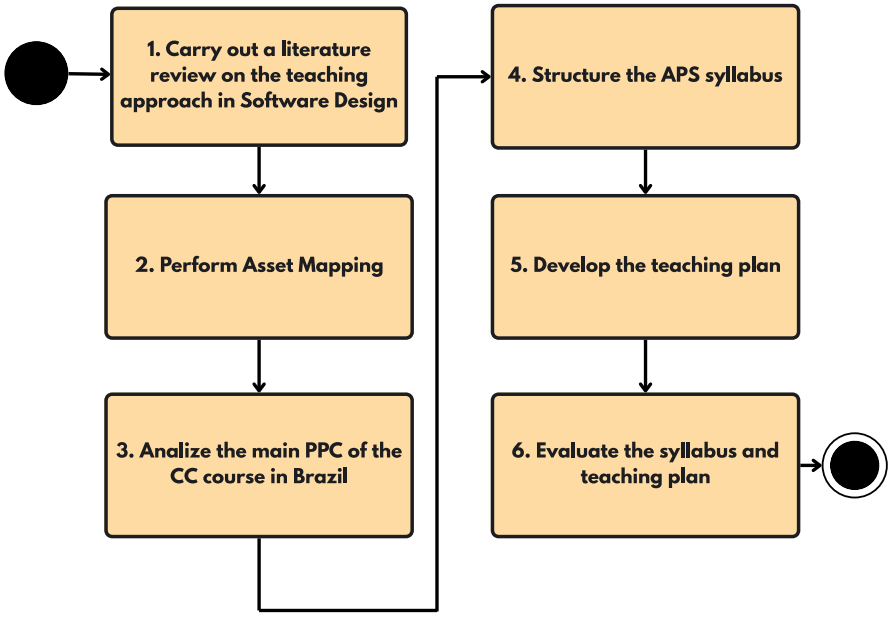

Figure 1. Research methodology
Source: Adapted from Castro and Oliveira (2024)

Subsections 2.1, 2.2, and 2.3 introduce steps 1, 2, and 3, respectively. Step 4 is presented in section 3, step 5 in section 4, and step 6 in section 7.

Literature Review on *Software Design* Teaching Approach

In step 1, a search was conducted in the specialized literature on the teaching of SD (*Software Design*) including: teaching approach (active or traditional); form of presentation of content (theoretical and/or practical); teaching strategies; forms of assessment; tools used; and group size for team activities (Castro and Oliveira, 2023b).

Regarding the teaching approach, it was identified that 90.20% of the works used the active form, which denotes that the use of active teaching methodologies should be applied to this discipline. Regarding the form of presentation of the contents, 54.90% of the studies presented teaching in a practical way and 35.29% in a theoretical and practical way, indicating that teaching in an active and practical way is presented with greater incidence in the specialized literature.

Another research question in the work of Castro and Oliveira (2023b) was: what teaching strategies were used in the teaching of *Software Design*? As results, the main strategies identified stand out: group project development; project-based learning; problem-based learning; hands-on laboratory; and flipped classroom. In the development of the course plan, Section 3, the main teaching strategies identified in the review were used.

In the literature review, the most cited forms of evaluation were those chosen to compose the course plan, however, for better monitoring of the project and *feedback* in the evaluation process, it was decided to include partial and final deliveries of the project, in addition to the traditional exercises and *individual quizzes*, which are important for the evaluation of theoretical contents. The oral presentation was also included as a form of evaluation, as it is mainly related to the flipped classroom strategy. Despite being the 5th most found evaluation option in the review, the use of a written test will not be used, because for the theoretical contents, exercises and quizzes will be used as a form of evaluation.

In terms of tools, dozens of tools used in the teaching of *Software Design were identified*. For the composition of the course plan, Astah was selected, which uses the *Unified Modeling Language* (UML) as the base language for the development of software projects.

Finally, given the higher incidence of works with a practical, active approach, which uses group teaching strategies, it was identified that the composition of these groups is a maximum of 6 students. In this sense, for group dynamics, the defined approach used this criterion as a maximum limiter of the size of the group of students in a given activity.

Mapping of Competencies between the ACM/IEEE and SBC Curricula for the Computer Science Course

The national curriculum guidelines for computing courses determine that the profile of the trainee must enunciate the desired competencies and skills (MEC, 2016). In line with this determination, the SBC presents the training references for undergraduate courses in computing based on competencies, whether generic, associated with the training axis, or derived competencies, which are related to specific contents (Zorzo *et al*., 2017).

In the international context, ACM/IEEE entities also define competency-based guidelines for computing courses (Curricula, 2020).

To identify the competencies related to SAD contents, step 2, the mapping of the assets was carried out between the SBC Training Frameworks for the Computer Science course (RF-CC-17) (Zorzo *et al*., 2017), the *Computing Curricula* 2020 (CC-2020) (Curricula, 2020) and the SWEBOK guide (Bourque and Fairley, 2014), which resulted in the publication of Castro and Oliveira (2022).

The mapping of competencies subsidized the development of the SAD course plan in directing the contents and skills necessary for the professional training of the graduate. In this sense, the following competencies from the SBC Training Frameworks (Zorzo *et al*., 2017) were selected for inclusion in the course plan proposal, considering as selection criteria the direct relationship with the software modeling process and adjacent areas that impact this process, such as the requirements process and the software construction process:

- Solve problems using programming environments (C1.3 and C.2.1);
- Recognize the importance of computational thinking in everyday life and its application in appropriate circumstances and in different domains (C.1.5);
- To design computational solutions from decisions, aiming at the balance of all factors involved (C.1.6. and C.4.7. - CE-VI);
- Apply recurring themes and principles, such as abstraction, complexity, *caching* principle, resource sharing, security, competition, systems evolution, among others, and recognize that these themes and principles are fundamental to the area of Computer Science (C.1.7.);
- Make decisions and innovate, based on knowledge of the operation and technical characteristics of hardware and software infrastructure of computer systems, aware of the ethical and legal aspects and the resulting environmental impacts (C.2.2.);
- Critically evaluate computer systems projects (C.2.3. and C.4.4. - CG-VIII);
- Employ methodologies that aim to ensure quality criteria throughout all stages of development of a computational solution (C.2.8. and C.4.8. - CE-VII);
- Analyze how much a computer-based system meets the criteria defined for its current and future use (suitability) (C.2.9. and C.3.9. - CE-VIII);
- Apply the principles of human-computer interaction to evaluate and build a wide variety of products including user interface, web pages, multimedia systems and mobile systems (C.2.11.);
- Identify and analyze requirements and specifications for specific problems and plan strategies for their solutions (C.3.8. - CE-IV).

To determine the contents necessary for the SAD discipline, the *Knowledge Area Software Design* (SD) of the SWEBOK Guide (Bourque and Fairley, 2014) was used as one of the axes, because in this guide there is a stratification of contents, which is not observed in RF-CC-17, which presents the contents without derivations. The SD, as well as the other *Knowledge Areas*, has the organization of content into *Topics* and for each Topic there are related *SubTopics*. The Subtopics have descriptions that specify which parts of this content should be observed, which was used for the compositions of the contents of the Teaching Units of the proposed course plan in SAD.

Analysis of the Main Pedagogical Project of the course (PPC) of the IFES in Brazil for the Computer Science Course (CC)

For the execution of step 3, the documentary analysis of the Pedagogical Projects of the Computer Science courses of 19 higher education institutions was carried out with the objective of identifying the aspects related to the *Software Design contents* in the PPC with the results published in (Castro and Oliveira, 2023a).

As a result of this research, it was observed the diversity of disciplines that deal with aspects of *Software Design* and the pulverization of these contents throughout the student's academic path, with the highest concentration in the 2nd and 3rd years of the course. Another relevant aspect was the finding that the contents related to *User Interface Design* are concentrated in the disciplines of Human-Computer Interface.

The aspects related to the Quality of Software *Design* are little explored in the CC courses, however in the proposed course plan for teaching for SAD they were included in Unit III, see section 3.4.

The analysis of the correlation between the subtopics of the SD subsidized the development of the Teaching Units, section 3, and the organization of the contents in order to deal with the subjects with greater correlation in the same Unit.

Table 1 presents the contents selected to compose the course plan for the teaching of SAD, which were the ones that had the highest incidence in the PPC analyzed. The criteria for the selection of the contents for the course plan were: compliance with the main Subtopics identified; and the inclusion of the Topics with the highest incidence.

Table 1. Contents selected to compose the course plan

Topic	Subtopic
Software Design Fundamentals	*General Design Concepts; Context of Software Design; Software Design Process; and Software Design Principles*
Software Struture and Architecture	*Architectural Structures and Viewpoints; and Families of Programs and Frameworks*
Software Design Quality Analysis and Evaluation	*Quality Attributes; and Quality Analysis and Evaluation Techniques*
Software Design Strategies and Methods	*Structural Descriptions; Behavioral Descriptions; General Strategies; Object-Oriented Design; Component-Based Design; and Other Methods*
Software Design Tools	*Software Design Tools*

The contents related to the Software *Design Quality Analysis and Evaluation Topic* were included in the plan due to the gap identified in the analysis of the PPC and the need for the graduate of this competency course.

The contents related to the *User Interface Topic* were not included in the plan, because in the mapping of the curricula and in the analysis of the PPC there was a concentration of the contents related to *the User Interface* in specific disciplines, such as Human-Computer Interface. In addition, these contents are not present in the Software Engineering discipline.

The Software Design Process *Subtopic* was included, as it showed a strong correlation with the Software *Design Fundamentals Subtopics*. In addition, *Software Design Tools* also showed a strong correlation with the Subtopics *Structural Descriptions* and *Behavioral Descriptions*, hence its inclusion.

Finally, the correlation analysis of the contents presented in the work by Castro and Oliveira (2023a) helped in the choices of which contents would be included in the Teaching Units of the proposed course plan.

COURSE PLAN

In this section, details regarding the structure of the course plan for the teaching of SAD will be presented. The structure used as a proposal of the definition of the teaching approach in other areas of Software Engineering served as reference works: in Software Testing (Elgrably and Oliveira, 2022); and in Software Processes (Quaresma and Oliveira, 2022)

It is also noteworthy that the course plan presented in Section 3.1 contemplated all aspects of its evaluation, described in Section 5, which is the final version of the course plan for the teaching of SAD.

Structure of the Course Plan

The structure of the course plan was obtained after an informal literature review on the key components present in course plans and the peer review process, Section 5. The structure for the elaboration of the plan for the teaching of SAD, after the peer review process, has the following components:

- Course objective: the key question for this component is why the course exists (Eberly et al., 2001; Matejka and Kurke, 1994)?
- Course prerequisites: what subjects must be taken by the student before taking the proposed course?
- Teaching units: how are the contents to be presented to the student during the course organization? The Teaching Units have the following components:
    - Prerequisites: which subjects must be taken by the student before the Teaching Unit?
    - Guiding questions: questions that will be answered during the teaching and learning process of the Unit;
    - Syllabus: topics of subjects related to the Teaching Unit;
    - Teaching strategies: approach used to present the content of the Teaching Unit;
    - Evaluation strategies: defines the evaluation format to be used in the Teaching Unit. The evaluation strategy will be related to the methodological procedure;
    - Tools: software that will be used during the execution of the Teaching Unit;
    - Expected results: what the student should be able to learn and accomplish after learning the Unit;
    - Learning Level: uses the terminology based on Bloom's Revised Taxonomy (Anderson and Krathwohl, 2001) and is related to the teaching strategies defined in the Unit that will be specified in methodological procedures.
- Expected outcomes: What should the student be able to learn and accomplish after the course? In addition, it defines what professional contributions to the student's education (Bowlick et al., 2020).

The basic structure of the course plan for teaching SAD, according to the components listed in this Section, is presented in Table 2.

Table 2. Basic structure of the course plan for teaching SAD

Objective
Present, using active learning strategies, the concepts related to the activity of Software Analysis and Design, as well as the design techniques and quality requirements
Prerequisites
Software Engineering, Database and Object-Oriented Programming
Teaching Units
I - Fundamentals of Software Design
II - Strategies for Software Design
III - Quality in Software Design
IV - Object-oriented Analysis and Design
Expected results
The student must be able to understand the concepts related to Software design. It should also situate analysis and design within the software life cycle, as well as design a software using appropriate techniques

The prerequisites indicated in Table 2 refer to fundamental contents for the student's understanding in this course. The Software Engineering discipline presents general content about the life cycle of a software, as well as the steps necessary for the production of a software. The contents of Database and Object-Oriented Programming are relevant, as they demonstrate how the implementation of the project will take place, helping the student in the cognitive process to design a software.

The Teaching Units were created with reference to the organization of the Topics of the SWEBOK guide (Bourque and Fairley, 2014) and the analysis of the PPC of the IFES, contemplating the contents presented in Table 1: Unit I is the contents referring to the Software *Design Fundamentals* Topic}; in Unit II were added the contents of the *Software Struture and Architecture, Software Design Strategies and Methods Topics* and *Software Design Tools*; in Unit III the contents of the Software *Design Quality Analysis and Evaluation Topic* were included; and in Unit IV was included the relationship of all Units with the objective of applying the contents in object-oriented software projects and the development of architectural documentation of the project.

In sections 3.2, 3.3, 3.4 and 3.5, the teaching units are presented, contemplating the objectives, their content structure and the main methodological procedures of the respective Unit. In addition, Section 3.6 presents the associated bibliographic references by Teaching Unit and contents, that can be used by the teacher for the development of didactic materials.

The methodological procedures describe the step-by-step process for carrying out the teaching practice, using the associated teaching methodology as a basis. In each methodological procedure, the following are presented: the type of activity (individual or group); the mechanical (how the procedure should be performed); evaluation criteria and strategies; and the level of learning expected according to Bloom's revised Taxonomy (Anderson and Krathwohl, 2001).

The selection of methodological procedures to compose the proposed course plan occurred according to the teaching strategies with greater incidence in the literature presented in Section 2.1. In addition, the authors of this work make use of these procedures in other areas of computer knowledge.

In all teaching units, the active teaching strategy dialogued expository class, was included, which, for Capraro (2007), is a technique that proposes to replace the lecture with advantages, in order to allow and favor the participation of the student, who always brings contributions, whether pertinent or not, to the class. In this teaching strategy, the teacher has the role of mediator so that students question, interpret and discuss the object of study (Coimbra, 2017).

The relationship between the methodological procedures and the contents should be carried out by the teacher in the construction of the teaching plan, that is, when there is an instance of the course plan in a class. The proposal for the selection of methodological procedures associated with the teaching units took into consideration the characteristics of contents of the unit.

The evaluation strategies are present in the composition of each Teaching Unit and in the specification of the methodological procedure. The evaluation strategies were selected according to their incidence in the specialized literature, as presented in Section 2.1.

Unit I - Fundamentals of Software Design

Unit I aims to present the main concepts about software projects, as well as to remind students about the stages of the software development process and the interface between Requirements Engineering and the APS stage. As it is the first Unit, subjects from prerequisite disciplines were added to its contents, mainly Software Engineering, as it broadly presents the structure of the base process for software development.

Table 3 presents Unit I - Fundamentals of Software Design, having the flipped classroom as a teaching strategy, as it is a strategy aimed at understanding concepts that are studied outside the classroom, presented and discussed by students in the classroom. The teaching strategy Problem-based Learning was also included in order to present the application of the concepts studied in the unit in practical exercises. In addition to the two strategies mentioned, the dialogued expository class was also included, given the need for the teacher to present the contents listed.

In terms of the level of learning following the definition of Bloom's revised Taxonomy (Anderson; Krathwohl, 2001), in Unit I the direction is for the basic levels of the cognitive dimension, that is, remembering and understanding.

Table 3. Unit I - Fundamentals of Software Design

Prerequisites
Software Engineering, Object-Oriented Programming and Algorithms
Guiding questions
What are the fundamental concepts involved in a software project?
How is the software project inserted in the Software life cycle?
What are the main concepts related to software architecture?
Content
1.1 Software engineering life cycle
1.2 Requirements Engineering
1.3 Principles of Software Design
1.4 Software Architecture Concept
1.5 Types of software projects

Table 3. Cont.

Teaching Strategies	**Evaluation Strategies**
Flipped Classroom and Dialogued Expository Class	Oral Presentation, Quiz and Exercises.
Learning Level	**Tools**
Remember/Conceptual Understand/Conceptual	ChatGPT Notion
Expected results	
The student must be able to understand the concepts related to Software design. It should also situate the analysis and design within the software life cycle.	

The levels of remembering and understanding were selected, as they are directly related to the type of content and the objectives defined for Unit I. The flipped classroom teaching strategy directs the student to meet these levels of learning, as the student will need to carry out studies outside the classroom environment, mainly related to fundamental concepts of software design.

As for the tools, the purpose of the indication of the ChatGPT tool was to include an Artificial Intelligence tool to support students in the process of conceptual research, with the teacher having the role of guiding students in how to use them. For the Notion tool, was indicated was as a tool for summarizing content, online collaboration and sharing.

Regarding the flipped classroom teaching strategy, defined in Table 3, two methodological procedures were developed for its execution, namely: Presentation and Discussion and *Fishbowl*.

Table 4 presents the methodological procedure regarding Presentation and Discussion. It is noted that the dynamics of the flipped classroom meet the aspect of group interaction, such as Presentation and Discussion, and the student's individualism, the *Fishbowl*.

Table 4. Methodological procedure - *Presentation and Discussion*

Name	**Type of Activity**
Presentation and Discussion	Group with 4 students
Mechanics	
Provide students with material on the topic of the flipped classroom.	
Divide students into groups of a maximum of 4 participants.	
Allow time for reading the materials and systematizing the information using the Notion tool.	
In the classroom presentation, the group must present the outline created in Notion and indicate some important questions about the topic.	
Presentation time must be a maximum of 15 minutes. The teacher must facilitate the organization of debate and student responses, in addition to encouraging discussion about concepts.	
Evaluation	**Learning level**
Criteria used to present the subjects: Mastery of the Theme; Presentation formatting; Questions produced; Oral Presentation and Delivery of the material produced.	Remember/Conceptual Understand/Conceptual

The methodological procedure defined in Table 5 presents an approach addressed to the development of a conversation among the students who are in the circle. Although the evaluation and the activity are characterized as individual, it is noted that the relevance of the dynamics will be in the interaction between students in order to feed the discussions in the center of the circle.

Table 5. Methodological procedure - Fishbowl

Name	**Type of Activity**
Fishbowl	Individual
Mechanics	
Provide students with material on the topic of the flipped classroom.	
Grant a deadline for reading the materials and systematizing the information using the Notion tool.	
In the classroom, the teacher should arrange the chairs in a circle with 4 chairs in the center of the circle. Three students should sit in the center of the circle, leaving an empty chair.	
The discussion on the topic of the flipped classroom must begin and within at least 5 minutes another student can enter the center of the circle. As soon as a new student sits down, one of the students who was in the discussion needs to stand up, always leaving the center with 3 students and 1 empty chair. This cycle is repeated until all students participate in the discussion.	
The teacher should mediate the discussion, including questions and encouraging students to participate in the dynamics.	
Evaluation	**Learning level**
Criteria used for the evaluation of each student: Mastery of the theme; participation in the dynamics; use of questions that were not addressed in the theme and associated;	Remember/Conceptual
Quiz in the next class about the concepts and discussions held; Oral Presentation and Quiz	Understand/Conceptual

Unit II - Strategies for Software Projects

Unit II - Strategies for Software Projects aims to present the forms of representation of a software project, as well as tools used in this process and the documentation of a software project. The inclusion of content on the types and specification of projects make Unit II concentrate a fundamental part of the software analysis process. This Unit will present UML, the language used for the development of modeling artifacts used for the documentation of a software at the project level.

Table 6 presents the structure of Unit II, covering the contents and teaching strategies.

The teaching strategies were selected according to the contents defined for the Unit, which mostly allow a practical and active application of these contents, such as items 2.3, 2.4, 2.5 and 2.6 of Table 6. Problem-based learning aims to stimulate the student's ability to research to develop solutions to a given problem and in the *hands-on laboratory strategy* the objective is for the student to have experience about the installation and use of modeling tools.

Table 6. Unit II - Strategies for Software projects

Prerequisites
Software Engineering, Requirements Engineering, Database, and Object-Oriented Programming
Guiding questions
What strategies are used to develop a software project?
How to model software?
How to specify and document the architecture of a software?
Content
2.1 Strategies for software design
2.2 Software Architecture Views: Object-Oriented Design, Component Design, Data Design, and Interface Design
2.3 Modeling tools: UML and CRC (Class Responsibility Card)
2.4 Standards-based design
2.5 Software Architecture Specification: Web, Mobile, and Service-Oriented Design
2.6 Software Design Documentation

Table 6. Cont.

Teaching Strategies	**Evaluation Strategies**
Dialogued expository class	
Problem-based learning	Oral Presentation, Quiz, Exercises and Final Delivery
Hands-on Laboratory	
Learning Level	**Tools**
Remember/Conceptual Understand/Conceptual	Astah
Apply/Procedural	Notion
Expected results	
The student must be able to apply the UML language to design a software.	
The student should be able to understand the main elements necessary for documentation on the software modeling stage. Also, use tools that support this process.	

Regarding the evaluation strategy using the Quiz, its use should preferably be for theoretical content, similar to the application in Unit I, see section 3.2. The use of the Quiz tool synchronously can bring greater interactivity with students, since the teacher of each question the teacher can explain the answer presented, before moving on to the next question, so that students can see why they got the question right or wrong.

Regarding the level of learning, the use of the Quiz makes it possible to remember and understand the themes and concepts seen by the student throughout the Unit. The level of learning to apply can be specifically addressed in the final delivery assessment strategies, as the student needs to perform a specific task by applying the knowledge in a new situation.

Still related to the level of learning, the teaching strategies problem-based learning and *hands-on* laboratory meet the apply level, as the student is directed to the application of the knowledge acquired in a specific scenario.

About the tools indicated for Unit II, Astah is a software that allows the construction of diagrams using UML. In addition, it has a free version that includes numerous UML diagrams that must be presented by the teacher. Notion, as mentioned earlier, helps the teacher in the provision of content and activities, and for students it enables the systematization of content and the development of shared activities.

To meet the practical contents of Unit II, three methodological procedures were planned, which are defined in Charts 7, 8 and 9. These procedures can be applied to content that deals with the use of modeling tools and development of software project documentation.

Table 7 presents the 24-hour dynamics as a specification of the Problem-based Learning teaching strategy. This procedure expects the student to solve a problem in a limited time, so the student will need concentration and organization with his partner to deliver a solution within the established deadline.

Table 7. Methodological procedure - 24-hour dynamics

Name	**Type of Activity**
Problem-based learning: 24h dynamics	Pair of 2 students
Mechanics	
Divide the students into pairs and present a problem on one of the subjects presented in the Unit.	
The students start the discussion about solving the problem in the classroom and if they have doubts they can call the teacher.	
The duo must organize the delivery within 24 hours after the presentation of the problem.	
Evaluation	**Learning level**
Criteria used for the evaluation of each student: Level of problem resolution; Formatting of the delivery; and Oral Presentation;	Apply/Procedural

The methodological procedure defined in Table 8 presents a pair activity for the development of UML diagrams, performed during class time. As it is a unit that requires the application of the contents, this procedure must be performed in the laboratory.

Table 8. Methodological procedure - Hands-on laboratory with UML

Name	**Type of Activity**
Hands-on Lab with UML	Pair of 2 students
Mechanics	
The teacher in the lab presents a use case diagram with documentation of the requirements and asks the students to develop other UML diagrams.	
In pairs, students should develop the requested diagrams and organize a delivery at the end of the class.	
Evaluation	**Learning level**
Criteria used for the evaluation of each student: Quality of the diagram and Final delivery	Apply/Procedural

Unlike the methodological procedure defined in Table 7, Table 9 presents a dynamics that begins in the classroom, with the presentation of the problem by the teacher and the students begin its resolution soon after. Because it deals with problems with a higher level of complexity, compared to those applied in the 24-hour dynamic, the deadline for final delivery should be between 3 and 5 days. The problems used by the teacher in this dynamics are mainly applied to item 2.3 of the content section of Table 6, referring to UML diagrams and CRC Cards, and content item 2.6 referring to software project documentation.

Table 9. Methodological procedure - Problem-based learning

Name	**Type of Activity**
Problem-based learning	Group with 4 students
Mechanics	
Division of the room into groups of 4 people. Presentation by the teacher of a problem related to the contents of the Unit.	
Students start negotiations to solve the problem in the classroom and have a deadline for final delivery of the activity between 3 and 5 days.	
After the delivery of the activity, the teacher promotes the discussion about the solution in the classroom and encourages the students to debate about possible solutions and doubts.	
Evaluation	**Learning level**
Criteria used to evaluate the group: Degree of compliance with the scope of the problem and Formatting of the material delivered; Forms of Evaluation: Final delivery	Apply/Procedural

Unit III - Quality in Software Design

After the presentation of the contents on APS, highlighting the context of the architecture and its application in different environments, Unit III aims to present the main models of maturity in software quality related to the APS process, as well as indicate the quality attributes associated with the software project.

Table 10 presents the structure of contents and the teaching strategies that were selected to deal with the theme of Quality in Software Design.

The contents of the unit were chained to foster the student in experimenting with the use of maturity models and the evaluation of a software project considering the quality attributes.

Unit III was included in the course plan to address the gap identified in Subsection 2.3, which pointed out that aspects related to the quality of software design are little explored in the Computer Science course.

Table 10. Unit III - Quality in software design

Prerequisites	
Software Engineering	
Guiding questions	
What are the fundamental requirements for a quality software project? How can a software project be evaluated in relation to quality?	
Content	
3.1 Quality attributes in software design 3.2 Maturity models in software engineering 3.3 Software project quality assessment	
Teaching Strategies	**Evaluation Strategies**
Dialogued expository class Problem-based learning	Oral Presentation, Exercises and Final Delivery.
Learning Level	**Tools**
Analyze/Procedural Assess/Metacognitive Create/Metacognitive	Checklist
Expected results	
The student should be able to understand the quality attributes associated with software design. The student must be able to perform quality assessment on a Software project.	

As one of the teaching strategies for Unit III, see Table 11, for the design of the methodological procedure Evaluation Consulting. This procedure will provide students with practical experience in the evaluation process of a software project. This dynamics will subsidize discussions that will be held in Unit IV, where students will develop a software project and will have to employ the quality requirements studied in this unit.

In the teaching strategy "Evaluation Consulting", observing the level of create/metacognitive learning, students need to analyze the problem scenario, evaluate which quality elements they will use and create a process that meets the given scenario.

Table 11. Methodological procedure - Evaluation Consulting

Name	**Type of Activity**
Problem-Based Learning: Evaluation Consulting	Group with 6 students
Mechanics	
Division in groups of 6 students. Professor presents a problem about a Software Quality consulting company that needs an evaluation model for Software design. Students must produce a software process, as well as an evaluation model with criteria/checklist for the company to use in their work. The teacher will play the role of client and the groups must organize themselves to deliver the solution at the end of the class. The teacher should organize iterations and make changes among team members during the execution of the activity, change the time of the iteration and other changes that encourage the adaptation of the group. The purpose of these changes is to enable the group to adapt over time to the changes. The group should present the resolution of the problem in class. The teacher will mediate a discussion about the works and encourage the participation of students in the discussion. At the end, the teacher should promote a retrospective of the activity, identifying important points of the dynamics and learning with the students.	
Evaluation	**Learning level**
Criteria used to evaluate the group: Quality of the process delivered; Participation in the Discussion; Formatting of the final delivery;	Create/Metacognitive

In addition to the teaching strategy related to Evaluation Consulting, see Table 11, Unit III will also use the 24-hour dynamics teaching strategy, defined in Table 7. The 24-hour dynamic, in the context of Unit III, must meet the levels of learning to analyze and evaluate. In this sense, the formulation of the dynamics can consist of a problem that students' evaluate in a software project document in order to identify strengths, weaknesses and opportunities for improvement.

The use of the *checklist* tool in this unit is justified by being an instrument for evaluation in software projects. Each item on the *checklist* can be a criterion to be met in the project, ensuring the conformity of the documentation being evaluated, fostering the student's thinking for the use of evaluation criteria before the production of the artifact and inducing the development of the software project in accordance with the established criteria.

In terms of evaluation strategies, the use of the final delivery, mainly related to the teaching strategy of the evaluation consultancy and the exercises to meet the contents on quality attributes in software design, were selected to contemplate unit III.

Unit IV - Object-Oriented Analysis and Design

Unit IV contains the contents related to Software Analysis and Design. This Unit aims to provide the student with experience in a project with real needs, that is, the requirements reflect the needs and experiences of the organization. In addition, this Unit will provide the student with practical activities related to object-oriented project modeling from the use of UML and development of software project documentation.

Table 12 presents the structure of contents, teaching and evaluation strategies that will be used during this Unit. The level of learning Create/Metacognitive is highlighted according to the student's need, beyond a real problem scenario, to propose a software project solution in an organized and documented way.

Table 12. Unit IV - Object-oriented Analysis and Design

Prerequisites	
Software Engineering, Object-Oriented Programming and Database	
Guiding questions	
How to build an object-oriented software project using UML modeling and develop consolidated project documentation?	
Content	
4.1 OO Project Modeling Using UML 4.2 Software Design Document Development	
Teaching Strategies	**Evaluation Strategies**
Dialogued expository class Project-based learning *Hands-on Laboratory*	Partial Project Delivery and Final Project Delivery
Learning Level	**Tools**
Create/Metacognitive	Astah Notion
Expected results	
The student must be able to develop and document a software project meeting the quality requirements, making use of UML.	

In Unit IV two methodological procedures were foreseen, namely: *Hands-on Laboratory*: object-oriented project with UML and Project with real client.

Table 13 was defined as a group type activity with 2 students so that communication is direct between the pair, so that the teacher can assess whether the competencies foreseen in the course have been acquired. In this teaching strategy, the evaluation chosen was the final delivery, because once the iterations of the pair are over, an artifact related to SAD must be delivered.

Table 13. Methodological procedure - Object-Oriented Design with UML

Name	**Type of Activity**
Hands-on Lab: Object-Oriented Design with UML	Pair of 2 students
Mechanics	
The teacher in the laboratory presents a requirements document and asks the students to develop UML diagrams that meet the OO project.	
The dynamics in the laboratory will consist of the work of the duo on a computer, performing 15-minute iterations to reverse the roles between the duo (pilot and co-pilot).	
The teacher at each iteration carries out a round of evaluation and doubts in order to monitor the progress of the execution and provide adjustments to the diagrams before the final version.	
Students should develop the requested diagrams and organize a delivery at the end of the class.	
Evaluation	**Learning level**
Criteria used for the evaluation of each group: Quality of the Diagrams.	Create/Metacognitive

For the methodological procedure for the development of a project with a real need, see Table 14, the group activity was defined for the student to have the experience of participating in this stage a group.

Table 14. Methodological procedure - Project with real need

Name	**Type of Activity**
Project-based learning: Project with real need	Group with 6 students
Mechanics	
The teacher will bring the demand for a software project from a local company or from the institution itself, so that the teacher is the central point of doubts regarding the project.	
The class will be divided into groups with 6 students to work on the project development.	
Students will be required to produce the software project documentation.	
There will be a partial delivery of the project, which will be evaluated by the teacher.	
At the end, the group must forward the documentation regarding the software architecture making use of UML and the quality requirements.	
Evaluation	**Learning level**
Criteria used for the evaluation of each group: Quality of the artifacts of partial delivery; Quality of the artifacts of the final delivery.	Create/Metacognitive

Bibliographic References for the Teaching Units

To assist the teacher in the process of preparing didactic materials related to each unit and its contents, Tables 15, 16, 17 and 18 present bibliographic references, as well as the indication of which chapters in these references are associated with each of the contents of the SAD course proposal.

Unit I, Section 3.2, as it deals with content referring to fundamentals and principles of software design, has references for all content in the book by Pressman and Maxim (2016) and Engholm (2010), see Table 15.

Table 15. References by contents of Unit I

Reference / Contents	1.1	1.2	1.3	1.4	1.5
Presman and Maxim (2016)	Chapters. 3 and 4	Chapter 8	Chapter 12	Chapter 13	Chapters 14 to 18
Sommerville (2011)	Chapter 2	Chapter 4	Chapter 5	Chapter 6	
de Pádua Paula Filho (2019)	Chapter 3	Chapter 4	Chapters 5 and 6	Chapter 6	
Engholm (2010)	Chapter 2	Chapter 3	Chapters 3 and 4	Chapters 3, 10 and 15	Chapter 11
Larman (2005)	Chapter 2	Chapters 4 and 5	Chapters 1 and 9	Chapter 13	
Budgen (2003)	Chapter. 3		Chapters 2, 5, 6 and 7		
Silveira et. al. (2011)			Chapters 3 and 4	Chapter 6	
Evans (2010)			Chapters 1 and 2	Chapters 4, 5 and 6	
Bass, Clemente e Kazman (2003)				Chapters 1 and 2	

Table 16 presents a set of references specifically for Unit II, Section 3.3. It is noted that it will be necessary to combine references to meet all the contents provided for Unit II. Fowler (2014), Guedes (2018) and Larman (2005) stand out as an important bibliographic reference for the unit, which together address the contents related to UML.

Table 16. References by contents of Unit II

Reference / Contents	2.1	2.2	2.3	2.4	2.5	2.6
Presman and Maxim (2016)		Chapters 13, 14 and 15	Chapter 10	Chapter 16	Chapters 17, 18 and 19	
Sommerville (2011)		Chapter 17	Chapter 7	Chapter 7	Chapters 18 and 19	
de Pádua Paula Filho (2019)		Chapter 10	Chapters 2 and 9		Chapter 6	Chapter 6
Engholm (2010)		Chapter 4	Chapter 9	Chapters 11 and 15	Chapter 11	Chapter 11
Larman (2005)		Chapters 14, 17 and 18	Chapters 6, 10, 15, 16, 28, 29, 37 and 39	Chapters 17, 25 and 26		Chapters 22 and 39
Budgen (2003)	Chapters 5 to 9	Chapters 16 and 17		Chapter 10		
Evans (2010)		Chapters 4 to 7		Chapters 4, 5 and 6		
Bass, Clemente e Kazman (2003)				Chapters 1 and 2		Chapter 18
Fowler (2014)			All chaps.			
Guedes (2018)			All chaps.			

For Unit III, see Section 3.4, the bibliographic references of Pressman and Maxin (2016) and Koscianski and Soares (2007) have all the contents that meet the unit, see Table 17. However, the MPS.BR General Guide for Software (SOFTEX, 2023), Chaudhary and Chopra (2016) and Chrissis, Konrad and Shrum (2011) are important as references for the unit, as they specifically present the processes and process quality attributes for the APS area according to the MPS.BR (Brazilian Software Process Improvement) and CMMI (*Capability Maturity Model Integration) maturity* models.

Table 17. References by contents of Unit III

Reference / Contents	3.1	3.2	3.3
Presman and Maxim (2016)	Chapters 12 and 19	Chapter 21	Chapters 13 and 20
Sommerville (2011)		Chapter 26	
de Pádua Paula Filho (2019)		Chapter 2	Chapter 3
Engholm (2010)		Chapter 17	
Budgen (2003)	Chapter 4		
Bass, Clemente e Kazman (2003)	Chapters 4 to 14		
Koscianski and Soares (2007)	Chapter 11	Chapters 5 and 6	Chapter 12
Chaudhary and Chopra (2017)		Chapters 1 and 2	
Chrissis, Konrad and Shrum (2011)		Chapters 1 and part 2	
SOFTEX (2023)		Chapters 8 and 9	

Finally, Table 18 presents the references that can be used by the teacher for the development of didactic materials specifically for the contents of Unit IV.

Table 18. References by contents of Unit IV

Reference / Contents	4.1	4.2
Presman and Maxim (2016)		Chapter 10
Sommerville (2011)	Chapter 7	
de Pádua Paula Filho (2019)	Chapters 2 and 9	
Engholm (2010)	Chapter 9	
Larman (2005)	Chapters 6, 10, 15, 16, 28, 29, 37 and 39	
Fowler (2014)	All chaps.	
Guedes (2018)	All chaps.	

TEACHING PLAN

The teaching plan is a communication tool between the teacher and the students and must be presented on the first day of class so that there is a discussion about the entire course planning (Gil, 2020).

The use of visual forms for the development of the teaching plan are alternatives for retaining information (Yarosh, 2021). In Kaur (2021), the use of the teaching plan in infographic format positively impacted the frequency of use and the preference of the infographic format over the traditional text format.

For the development of the teaching plan referring to the course plan presented in Section 3, the format of infographic 1 will be adopted, contemplating the following information:

- **Information about the course[1]:** presentation of the objective of the course;
- **Expected results:** expectation of knowledge achieved at the end of the course;
- **Technologies used:** tools used for administration/communication of the discipline;

1 https://zenodo.org/records/11166196

- **Definitions of the Units:** inclusion of the four teaching units, as well as the guiding questions;
- **Schedule:** definition of the activities and contents to be dealt with on the respective date;
- **Evaluations:** definition of the evaluation format and the respective score for the composition of the final grade;
- **Teaching strategies:** definition of the teaching strategies that will be used for the teaching-learning process; and
- **Complementary links:** free access links for students to the digital bibliographic collection and spreadsheet for monitoring grades and attendance.

The teaching plan in infographic format, Figure 2, is a proposal for a teaching plan based on the course plan specified in this work and the calendar for the second semester of the year 2023 of the CC course at the Federal University of Pará.

Figure 2. Representation of the teaching plan in infographic
Source: Elaborated by the authors

It is also noteworthy that the teaching plan presented in this Section contemplated all aspects of its evaluation, described in Section 6, which is the final version of the teaching plan.

One of the elements used in the Important Links section was the inclusion of QR-Code to redirect to the indicated site. In addition, the use of icons for evaluations and their association in the schedule allows the student to visualize the moments of dynamics in the classroom, Quiz, Exercises and Project delivery.

EVALUATION OF THE COURSE PLAN AND TEACHING PLAN

The development of the instruments, course plan and teaching plan, were submitted to the peer review process, which is a collaborative process that allows experts to evaluate the work and provide suggestions and improvements (Camargos, 2018). In addition, Camargos (2018) points to speed, professionalism, collaboration, and deadline as fundamental characteristics for the peer review process.

The peer review process followed the steps: Identification of Reviewers; Development of the Review Form; Initial Review of the Work; Adjustments/Correction of the Indicated Items; and Final Review (Castro and Oliveira 2015).

The first step in the peer review process was the identification of the reviewers. To this end, an e-mail was sent with the necessary information for the review process, as well as the invitation to the participants of the peer review process. In total, three evaluators were selected, and named A1 - Evaluator 1, A2 - Evaluator 2 and A3 - Evaluator 3.

The process of selection of evaluators was conducted by the professor who supervised this work based on his research network, which involves professors from other public and private educational institutions in Brazil.

Table 19 presents a brief description of the evaluators' curriculum, as well as the areas of expertise in software engineering.

Table 19. Profile of the Evaluators selected in the Peer Review process

Appraiser	Profile
A1	PhD and Post-Doctorate in Computer Science. Evaluator and Instructor of the MPS.BR quality model for software development. He has experience in the area of Software Engineering, Software Quality and Educational Informatics.
A2	PhD in Computer Science with a research line in Software Engineering. He has experience in the use of active methods in the area of Software Engineering.
A3	PhD in Computer Science. He has academic experience in the area of Software Engineering, Project Management, Process Modeling, Software Process Quality Models. In addition, it conducts research on the Teaching-Learning of Software Engineering.

The development of the review form was based on the criteria defined in the work of Castro and Oliveira (2015) contemplating the items:

- TA - High technical: indicating that a problem has been found in an item that, if not changed, will compromise the considerations;
- TB - Low Technical: indicating that a problem has been found in an item that would be convenient to change;
- E - Editorial: indicating that a Portuguese error was found or that the text needs to be improved;

- Q - Questioning: indicating that there was doubt as to the content of the considerations;
- G - General: indicating that the comment is general in relation to the considerations.

The review form presented was an electronic spreadsheet with two columns: the first with the criterion, that is, for the evaluator to select - TA, TB, E, Q or G; and the second column to place the observation/comment associated with the selected criterion.

The selected evaluators were invited to participate in *online* meetings to present the course plan and teaching plan, as well as explanations about the review form. Due to the number of components of the instruments, 4 meetings were necessary to fully present all the elements of the course plan and teaching plan.

After the presentations, the evaluators filled out the evaluation form and forwarded it to the researchers. The first submission of the evaluation forms was considered as the initial Review step of the work. Table 20 presents the compilation with all the indications made by the evaluators in the initial review of the work.

Table 20. Consolidation of peer review forms

Appraiser	Criterion	Description
A1	TA	Include the level of learning of the revised Bloom taxonomy (cognitive dimension and knowledge dimension) in the methodological procedures indicated for each specified unit.
A1	TB	Remove course workload as a field of the teaching approach, as there are no elements to determine the workload foreseen for the course plan.
A1	TA	Specify the mechanics for the execution of the methodological procedures defined in each teaching unit.
A1	TB	Describe the purpose of use for each tool mentioned in the teaching plan.
A2	E	Specify the items that are components in the teaching unit in order to make it clear to the reader what the objective of the given field is.
A2	E	Describe how to use the tools, especially ChatGPT. I suggest that the teacher present to the students a way to use the tool.
A2	TB	Describe a brief summary of the active methodology that will be employed in the teaching plan.
A3	TB	Remove the fields related to the course plan from the structure of the approach, namely: course schedule; student dishonesty policy; grade policy for the course; use of technologies; common mistakes; how to succeed in the course. I recommend that these fields may be included in the format of the teaching plan (instance of the course plan to be applied when performing an experiment), given the relevance of the information for the execution of the approach.
A3	G	Define at chapter level the references suggested by teaching unit.
A3	TB	In the teaching plan, make it clear in the execution schedule at which times there will be evaluation and what format it will be.

After all the adjustments indicated were made in the initial review of the study, the final version of the instruments was sent to the evaluators, including all the corrections indicated in Table 20. There was no request for change in the final version demanded by the evaluators, ending the peer review process.

Analyzing the items indicated in Table 20, it is observed that most of the indications for adjustments in the proposal were in relation to the use of Bloom's revised taxonomy, cited for use both in the general structure of the teaching units and for the methodological procedures. It is also noted that there was no indication as to the composition of the contents contained in the units, which is a positive aspect in the course plan elaborated.

Finally, regarding the items indicated in Table 20 referring to the teaching plan, it is noted that all indicated improvements in order to specify the elements indicated, such as the tools, the active methodologies and the schedule.

DISCUSSION OF THE RESULTS

In this section, the analysis and discussions on the results obtained by the development of the course plan and teaching plan will be presented.

The first important result of the structure of the course plan for SAD is the concentration of the main contents on Software Analysis and Design, obtained from the SWEBOK guide (Bourque and Fairley, 2014) and the analysis of the PPC of Computer Science courses in Brazil, see Section 2.3, with the objective of providing students with better professional training in the area of computing in emerging topics (Ferreira *et al*., 2018; Forti *et al*., 2022).

The use of active methodologies was included in the course plan, as it is an approach that is a trend in teaching and these were identified in the results presented in Section 2.1 and in the curricular guidelines for computing courses (Force, 2020; Zorzo *et al*., 2017).

The aspects related to the interface project were not contemplated in the contents in detail in the course plan, and only general presentations on the subject were included in the proposal. According to the work of Castro and Oliveira (2023a), these contents related to interface design are concentrated exclusively in the disciplines of Human-Computer Interaction.

The inclusion of the methodological procedure "Project in a local company", see Table 10, seeks to involve students with the market, making them interact with real problems and present a vision of what solutions can be applied in that context.

Regarding the organization of the methodological procedures, it is noted that activities were defined in groups with a maximum of 6 people, justified by the need for the student to act as a team, foster discussion, division of tasks and strengthen student engagement.

The theme of quality, specifically in SAD, was contemplated in the proposal in Unit III, see Section 3.4. This theme is present in the SWEBOK guide (Bourque and Fairley, 2014) and absent in most of the disciplines analyzed in Section 2.3, which motivated its inclusion because it is understood that quality aspects are fundamental for the evolution of the software project.

Regarding the teaching plan, an instance of the course plan, there is innovation in presenting the elements arranged in an infographic format. The use of iconography and QR-Code in the document is presented as visual resources that facilitate the interpretation of the teaching plan and use. In addition, in the evaluation of the teaching plan, Section 5, points of improvement were indicated and included in the version presented in this article.

RELATED WORKS

In this section, some related works will be presented, as well as the differentiation and relationships with this work.

In the work of Quaresma and Oliveira (2022), a course plan was developed for the area of Software Process. In addition, for the elaboration of the course plan, an analysis was carried out in the literature on the main subjects dealt with in software process. Unlike Quaresma and Oliveira (2022), this work addresses the knowledge area of Software Analysis and Design and made use of documentary analysis of the Pedagogical Projects of Computer Science courses in Brazil and teaching approaches identified in the literature as a justification for inclusion in the course plan.

In the work of Elgrably and Oliveira (2020), a course plan was created for the teaching of Software Testing. In terms of research methodology, there is similarity with this work, however regarding the structuring of the teaching units, in this work there was the inclusion of teaching strategies, evaluation strategies and indicated tools.

The work by Bowlick, Bednarz and Goldberg (2020) deals with a research in a course plan aimed at identifying the panorama of the teaching of Geographic Information Systems (GIS) programming. Despite dealing with a research in a course plan, the work of Bowlick, Bednarz and Goldberg (2020) does not aim to develop a course plan for the teaching of SAD, which differs from this work.

Regarding the teaching plan, the work by Rubio et al. (2022) presents relevant aspects that should be included in the teaching plan in the area of Mechanical Engineering. In addition, the emphasis of the work directs to constant adaptations in the teaching plan, as well as the inclusion of icons, images and graphics, which aims to substantially improve the content and understanding of the course syllabus by the students.

CONCLUSION

This paper presented a proposal for a course plan for the teaching of SAD for the Computer Science course using active methodologies for the development of contents. This proposal includes four teaching units that are interrelated to meet the competencies necessary for the learners to perform their role as graduate students of the course.

In addition, the inclusion of a specific teaching unit for Quality in Software Design, see Section 3.4, fills an existing gap in the PPC of the Computer Science courses analyzed, making this aspect one of the pillars for the development of the analysis and design activity of a software.

As teaching strategies, different active methodologies were added to each unit, in order to present the contents in different ways with the aim of bringing greater dynamics to the teaching and learning process.

Another highlight was the process of evaluation of the course plan, see Section 4, which indicated points of improvement for the proposed course plan, being essential for the evaluation of this proposal by a group of specialists in the field of Software Engineering, which worked as a risk mitigation mechanism of the referred plan.

The presentation of the teaching plan in infographic format allows a better visualization of the information about the execution of the course, as well as bringing together all the information in a single document.

The absence of related works dealing with the development of curriculum for the discipline of Software Analysis and Design, makes the work innovative that contributes to research in the area of computer education.

As limitations of the work, the contents selected for the composition in the teaching units are exclusive to the SWEBOK guide and the analysis of the PPC considered only the Computer Science courses with grade 5 of the ENADE.

For future works, it is intended to carry out an experiment in an APS class of the Computer Science course in order to observe the impacts on the teaching-learning process of the instruments presented, as well as the inclusion of new contents in the training of students.

REFERENCES

Anderson, L. W. and Krathwohl, D. R. (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. Longman.

Aniche, M., Yoder, J., and Kon, F. (2019). Current challenges in practical object-oriented software design. In 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), pages 113–116.

Bass, L., Clements, P., and Kazman, R. (2003). Software architecture in practice. Addison-Wesley Professional.

Bourque, P. and Fairley, R. E., editors (2014). SWEBOK: Guide to the Software Engineering Body of Knowledge. IEEE Computer Society, Los Alamitos, CA, 3.0 edition.

Bowlick, F. J., Bednarz, S. W., and Goldberg, D. W. (2020). Course syllabi in gis programming: Trends and patterns in the integration of computer science and programming. The Canadian Geographer/Le Geógraphe canadien, 64(4):495–511.

Budgen, D. (2003). Software design. Addison-Wesley.

Camargos, E. F. (2018). Peer review: importance, responsibilities, and benefits. Geriatrics, Gerontology and Aging, 12(3):141–142.

Capraro, L. (2007). Técnicas de ensino a serviço do professor engenheiro. In XXXV Congresso Brasileiro de Educação em Engenharia, COBENGE, Curitiba-PR.

Castro, V. d. S. e Oliveira, S. R. B. (2015). Um framework de práticas Ágeis para apoio à implementação do processo de projeto e construção do produto. iSys – Brazilian Journal of Information Systems, 8(2): 78–97.

Castro, V. d. S. and Oliveira, S. R. B. (2022). Content and competences for teaching software design in the computer science course: A mapping of CC-2020, RF-CC-2017 and SWEBOK-v3.0. 19th CONTECSI - INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGY MANAGEMENT VIRTUAL.

Castro, V. d. S. and Oliveira, S. R. B. (2023a). A diagnosis on the teaching of software design in a sample of undergraduate courses in computer science in brazil. In 2023 IEEE Frontiers in Education Conference (FIE).

Castro, V. d. S. and Oliveira, S. R. B. (2023b). Diversity in software design and construction teaching: A systematic literature review. Education Sciences, 13(3):303.

Castro, V. d. S. and Oliveira, S. R. B. (2024). Software Analysis and Design: A course plan using active teaching methods in Computer Science Course. 20th CONTECSI - INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGY MANAGEMENT VIRTUAL.

Chaudhary, M. and Chopra, A. (2016). CMMI for development: Implementation guide. Apress.

Chaudhary, M. and Chopra, A. (2017). CMMI for Development. Apress.

Chrissis, M. B., Konrad, M., and Shrum, S. (2011). CMMI for development: guidelines for process integration and product improvement. Pearson Education.

Coimbra, C. L. (2017). A aula expositiva dialogada em uma perspectiva freireana. LEAL, Edvalda Araújo; MIRANDA, Gilberto José; CASA NOVA, Silvia Pereira de Castro. Revolucionando a Sala de Aula: como envolver o estudante aplicando técnicas de metodologias ativas de aprendizagem. São Paulo: Atlas, pages 1–13.

Curricula, C. (2020). Paradigms for global computing education. URL: https://dl.acm.org/doi/book/10.1145/3467967.

de Pádua Paula Filho, W. (2019). Engenharia de Software: produtos. LTC, Rio de Janeiro.

de Sena Quaresma, J. A. e Oliveira, S. R. B. (2022). Evaluation and use of a student- centered syllabus for the software process subject in a postgraduate course: A quasi- experiment. Education Sciences, 12(12):851.

Eberly, M. B., Newton, S. E., and Wiggins, R. A. (2001). The syllabus as a tool for student-centered learning. The Journal of General Education, pages 56–74.

Elgrably, I. S. and Bezerra Oliveira, S. R. (2022). A quasi-experimental evaluation of teaching software testing in software quality assurance subject during a post-graduate computer science course. International Journal of Emerging Technologies in Learning, 17(5).

Elgrably, I. S. and Oliveira, S. R. B. (2020). Construction of a syllabus adhering to the teaching of software testing using agile practices. In 2020 IEEE Frontiers in Education Conference (FIE), pages 1–9, Uppsala, Sweden. IEEE.

Engholm, H. (2010). Engenharia de software na prática. Novatec Editora, São Paulo, Brasil.

Evans, E. (2010). Domain-Driven Design: Atacando as complexidades no coração do software. Alta Books.

Ferreira, T., Viana, D., Fernandes, J., and Santos, R. (2018). Identifying emerging topics and difficulties in software engineering education in brazil. In Proceedings of the XXXII Brazilian Symposium on Software Engineering, pages 230–239. Association for Computing Machinery.

Flauzino, R. H., Peres, C. M., and Carmona, F. (2021). A descoberta do plano de Ensino e aprendizagem (PEA) como instrumento reflexivo na docência. Medicina (Ribeirão Preto)

Force, C. T. (2020). Computing Curricula 2020. ACM.

Forti, S., Breitenbucher, U., and Soldani, J. (2022). Trending topics in software engineering. SIGSOFT Softw. Eng. Notes, 47(3):20–21.

Fowler, M. (2014). UML Essencial: um breve guia para linguagem padrão. Bookman editora.

Garousi, V., Giray, G., and Tuzun, E. (2019). Understanding the knowledge gaps of software engineers: An empirical analysis based on swebok. ACM Trans. Comput. Educ., 20(1).

Gil, A. C. (2000). Metodologia Do Ensino Superior. Editora Atlas SA.

Guedes, G. T. (2018). UML 2-Uma abordagem prática. Novatec Editora.

Kaur, A. W. (2021). "dope syllabus": Student impressions of an infographic-style visual syllabus. International Journal for the Scholarship of Teaching and Learning, 15(2):6.

Koscianski, A. and dos Santos Soares, M. (2007). Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. Novatec Editora.

Larman, C. (2005). Utilizando UML e Padrões, 3ª edição ed. Bookman.

Leite, F. T., Coutinho, J. C. S., and de Sousa, R. R. (2020). An experience report about challenges of software engineering as a second cycle course. In Proceedings of the XXXIV Brazilian Symposium on Software Engineering, SBES '20, page 824–833, New York, NY, USA. Association for Computing Machinery.

Lopes, A. C. (2014). Teorias de currículo. Cortez Editora.

Matejka, K. and Kurke, L. B. (1994). Designing a great syllabus. College Teaching, 42(3):115–117.

MEC (2016). Resolução nº 05, de 16 de novembro de 2016, diretrizes curriculares nacionais para os cursos de graduação em computação. Technical report, Ministério da Educação - Brasil.

on Computing Curricula, J. T. F. (2013). Computer Science Curricula 2013. ACM/Association for Computing Machinery.

Padilha, P. R. (2002). Planejamento dialógico: como construir o projeto político-pedagógico da escola. Cortez/Instituto Paulo Freire.

Pressman, R. S. and Maxim, B. R. (2016). Engenharia de software. McGraw Hill Brasil, Porto Alegre.

Quaresma, J. A. S. and Oliveira, S. R. B. (2022). A syllabus proposal for teaching of software development process in undergraduate courses in computer science. In Proceedings of the XXXVI Brazilian Symposium on Software Engineering, pages 153–167.

Rodríguez-Pérez, G., Nadri, R., and Nagappan, M. (2021). Perceived diversity in software engineering: a systematic literature review. Empirical Software Engineering, 26(5):1– 38.

Rubio, F., Llopis-Albert, C., and Zeng, S. (2022). Best practices in syllabus design and course planning applied to mechanical engineering subjects. Multidisciplinary Journal for Education, Social and Technological Sciences, 9(2):123–137

Sacristán, J. G. (2013). O que significa o currículo. Saberes e incertezas sobre o currículo. Porto Alegre: Penso, pages 16–35.

Silveira, P., Silveira, G., Lopes, S., Moreira, G., STEPAAT, N., and Kung, F. (2011). Introdução à Arquitetura de Design de Software: Uma Introdução à Plataforma Java. Elsevier Brasil.

SOFTEX (2023). MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral MPS de Software. Softex.

Sommerville, I. (2011). Engenharia de software. Pearson Prentice Hall, São Paulo.

Yarosh, J. H. (2021). The syllabus reconstructed: an analysis of traditional and visual syllabi for information retention and inclusiveness. Teaching Sociology, 49(2):173–183.

Zabeu, A. C., Rocha, A. R., Ângela Filipak Machado, C., dos Santos Souza, G., and Reinehr, S. (2021). MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral MPS de Software. Softex.

Zorzo, A. F., Nunes, D., Matos, E. S., Steinmacher, I., Leite, J. C., Araujo, R., Correia, R. C. M., and Martins, S. (2017). Referenciais de Formação para os Cursos de Graduação em Computação. SBC.

Editor-in-chief: Edson Luiz Riccio

Data Availability Statement: All data generated or analysed during this study are included in this published article.