JPB REVIEW
INTERNATIONAL JOURNAL OF PROFESSIONAL BUSINESS REVIEW
ISSN: 2525-3654
OPEN ACCESS

PROPOSAL OF A SYSTEMIC APPROACH FOR ACADEMIC SOFTWARE: JOINT METHODOLOGY FOR EDUCATIONAL APPLICATIONS (JMEA)

Daniel Benito Moran[A], Anabelem Soberanes Martín[B], José Luis Castillo Mendoza[C]

ARTICLE INFO

Article history:

Received: Jun, 03rd 2024

Accepted: Aug, 02nd 2024

Keywords:

Educational Software;
Educational Technology;
Methodology;
Systems Design;
Educational Methods.

ABSTRACT

Objective: This study presents a proposed methodology for the development of educational software. This methodology will serve as a guide for development teams that need to integrate the academic context with technological elements and vice versa.

Theoretical Framework: The study methodology is grounded in theories that advocate for the flexibility necessary in educational settings, moving away from traditional software development approaches. The theoretical contributions include constructivism, which stresses the importance of adaptable, learner-focused methods for effective educational software.

Method: The study utilized applied research to examine the integration of educational approaches within software engineering processes. The methodology included a combination of expert validation and practical application within development teams, ensuring the proposed methods were both theoretically sound and practically viable.

Results and Discussion: The developed methodology merges the software development lifecycle with educational processes, addressing adaptative and non-linear nature of academic contexts. Expert validation and team application demonstrated the methodology's effectiveness in creating functional educational software aligning with learning objectives.

Research Implications: This study contributes to educational software development by providing a structured methodology that integrates pedagogical considerations throughout the process development. It highlights the importance of adaptability and educational alignment in software design, offering significant implications for software engineering and academic methodology.

Originality/Value: The originality of this research lies in its structured approach to combining educational theories and software development practices, addressing a gap in current methodologies. It provides a framework for developing educational software that is technically robust and pedagogically effective, enhancing the utility of educational technologies in real-world learning environments.

Doi: https://doi.org/10.26668/businessreview/2024.v9i9.4848

PROPOSTA DE ABORDAGEM SISTÊMICA PARA SOFTWARE EDUCACIONAL: METODOLOGIA CONJUNTA PARA APLICAÇÕES EDUCACIONAIS (MCAE)

RESUMO
Objetivo: Este estudo apresenta uma proposta de metodologia para o desenvolvimento de software educacional. Esta metodologia servirá como guia para as equipes de desenvolvimento que precisam integrar o contexto acadêmico com elementos tecnológicos e vice-versa.

[A] *PhD in Computer Science. Tecnológico de Estudios Superiores de Chicoloapan. Chicoloapan de Juárez, Estado de México, Mexico. E-mail: daniel.benito@teschic.edu.mx Orcid: https://orcid.org/0000-0002-4015-8342*
[B] *PhD in Educational Science. Universidad Autónoma del Estado de México. Valle de Chalco, Estado de México, Mexico. E-mail: asoberanesm@uaemex.mx Orcid: https://orcid.org/0000-0002-1101-8279*
[C] *PhD in Educational Science. Universidad Autónoma del Estado de México. Valle de Chalco, Estado de México, Mexico. E-mail: jlcastillom@uaemex.mx Orcid: https://orcid.org/0000-0002-5668-0602*

Referencial Teórico: A metodologia do estudo é fundamentada em teorias que advogam pela flexibilidade necessária em ambientes educacionais, afastando-se das abordagens tradicionais de desenvolvimento de software. As contribuições teóricas incluem o construtivismo, que enfatiza a importância de métodos adaptáveis e centrados no aprendiz para um software educacional eficaz.
Método: O estudo utilizou pesquisa aplicada para examinar a integração de abordagens educacionais nos processos de engenharia de software. A metodologia incluiu uma combinação de validação por especialistas e aplicação prática dentro das equipes de desenvolvimento, garantindo que os métodos propostos fossem teoricamente sólidos e praticamente viáveis.
Resultados e Discussão: A metodologia desenvolvida mescla o ciclo de vida do desenvolvimento de software com processos educacionais, abordando a natureza adaptativa e não linear dos contextos acadêmicos. A validação por especialistas e a aplicação em equipe demonstraram a eficácia da metodologia na criação de software educacional funcional alinhado com os objetivos de aprendizagem.
Implicações da Pesquisa: Este estudo contribui para o desenvolvimento de software educacional fornecendo uma metodologia estruturada que integra considerações pedagógicas ao longo do processo de desenvolvimento. Destaca a importância da adaptabilidade e do alinhamento educacional no design de software, oferecendo implicações significativas para a engenharia de software e metodologia acadêmica.
Originalidade/Valor: A originalidade desta pesquisa reside em sua abordagem estruturada para combinar teorias educacionais e práticas de desenvolvimento de software, abordando uma lacuna nas metodologias atuais. Fornece um arcabouço para o desenvolvimento de software educacional que é tecnicamente robusto e pedagogicamente eficaz, melhorando a utilidade das tecnologias educacionais em ambientes de aprendizagem do mundo real.

Palavras-chave: Software Didático, Tecnologia Educativa, Metodología, Engenharia de Sistemas, Método de Ensino.

PROPUESTA DE UN ENFOQUE SISTÉMICO PARA SOFTWARE EDUCATIVO: METODOLOGÍA CONJUNTA PARA APLICACIONES EDUCATIVAS (MCAE)

RESUMEN

Objetivo: Este estudio presenta una propuesta de metodología para el desarrollo de software educativo. Esta metodología servirá como guía para los equipos de desarrollo que necesiten integrar el contexto académico con elementos tecnológicos y viceversa
Marco Teórico: La metodología del estudio se basa en teorías que abogan por la flexibilidad necesaria en entornos educativos, alejándose de los enfoques tradicionales de desarrollo de software. Las contribuciones teóricas incluyen el constructivismo, que subraya la importancia de métodos adaptables y centrados en el aprendiz para un software educativo efectivo.
Método: El estudio utilizó la investigación aplicada para examinar la integración de enfoques educativos dentro de los procesos de ingeniería de software. La metodología incluyó una combinación de validación por expertos y aplicación práctica dentro de los equipos de desarrollo, asegurando que los métodos propuestos fueran teóricamente sólidos y viables prácticamente.
Resultados y Discusión: La metodología desarrollada fusiona el ciclo de vida del desarrollo de software con los procesos educativos, abordando la naturaleza adaptativa y no lineal de los contextos académicos. La validación por expertos y la aplicación en equipo demostraron la efectividad de la metodología para crear software educativo funcional que se alinea con los objetivos de aprendizaje.
Implicaciones de la Investigación: Este estudio contribuye al desarrollo de software educativo proporcionando una metodología estructurada que integra consideraciones pedagógicas a lo largo del desarrollo del proceso. Destaca la importancia de la adaptabilidad y la alineación educativa en el diseño de software, ofreciendo implicaciones significativas para la ingeniería de software y la metodología académica.
Originalidad/Valor: La originalidad de esta investigación radica en su enfoque estructurado para combinar teorías educativas y prácticas de desarrollo de software, abordando una brecha en las metodologías actuales. Proporciona un marco para desarrollar software educativo que es técnicamente robusto y pedagógicamente efectivo, mejorando la utilidad de las tecnologías educativas en entornos de aprendizaje del mundo real.

Palabras clave: Software Didáctico, Tecnología Educativa, Metodología, Ingeniería de Sistemas, Método de Enseñanza.

1 INTRODUCTION

The teaching-learning process seeks to integrate theories that optimally support learning and teaching; Gimeno (2008) defines the latter as the technique that guides learning to consolidate objectives and establishes that the idea of teaching is the one in charge of systematically structuring learning, thus evidencing the relationship between these two concepts.

In this sense, teaching theories guide the elements involved in learning, so it is necessary to consider the various situations in which the educational process unfolds (Munna & Kalam, 2021; Gimeno, 2008); therefore, teaching must contain a solid theoretical foundation based on academic theories that make it possible to put it into practice, to consider problems and variables present during execution (Flores, 1997). Heredia & Sánchez (2013) define them as a set of interlinked constructs that observe, describe, and explain people's learning process and what is thought to be related.

To further enrich this foundation, Hinostroza et al. (2000) propose a professional tool perspective in developing educational software, where the software serves not only as a learning platform but also as an integral part of enhancing teachers' pedagogical practice. This approach highlights the necessity of involving educators in the software development process to ensure the tools are contextually relevant and pedagogically sound, offering a methodology that aligns with the professional needs and educational goals of teachers (Hinostroza et al., 2000).

Therefore, it is necessary to point out that Hammad et al. (2020), Schunck (2012), Urías et al. (2015), and Ríos (2001) refer to the following theories as representative of this area of study:

- Behaviorismo;
- sociocultural theory of Lev Semyonovich Vygotsky;
- cognitivism;
- constructivism;
- connectivism.

In this way, the theories seek to be applied in education since they tend to be dynamic and adaptive, so their evolution is inferred from environmental changes (Hammad et al., 2020; Pineda, 2017). On the other hand, in recent years, the educational context has used technology, which has been supported by learning theories and various disciplinary fields such as curriculum, communication, and systems. Although these theories contribute elements of significance, integrating all the disciplinary fields often leads to the expected result (Mohebi, 2021). The objective of this research is to present the proposal for a methodology for the development of educational software that serves as a guide for development teams that require the integration of the academic context with technological elements and vice versa, as well as its validation using some techniques that support the construction of JMEA.

2 THEORETICAL FRAMEWORK

In the specialized domain, employing our methodologies and methods for developing computer systems imbues them with an engineering essence, as these practices facilitate systematic and reproducible tasks aimed at fulfilling user requirements and achieving predetermined goals. Moreover, this approach offers valuable perspectives and tools, including coping strategies, notation, and procedures, for addressing the problem; consequently, it is essential to reconsider or adjust these strategies in nonlinear contexts (Pineda, 2017; Knippers et al., 2021). Liviu (2014) defines a methodology for software development as a series of rules and guidelines used during the research, planning, design, development, testing, configuration, and maintenance of software. It also includes core values applicable to the development team and tools used throughout the process.

The definition elucidates that all activities are primarily directed towards fulfilling the technical aspects of outcomes, which is predominantly effective in automated processes where variables remain relatively static. Nonetheless, various factors can complicate achieving objectives in scenarios where processes are dynamic, such as in educational settings (Žužek et al., 2020; Abud, 2009). As a result, numerous models and methodologies have been devised over time to address these challenges and attain the specified learning objectives. Here are some of them:

- educational software engineering (Galvis, 1992);
- model of analysis, instructional, technological design, and evaluation (ADITE in Spanish *Modelo de análisis, diseño instruccional, tecnológico y evaluación*) (Pole, 2003);
- object oriented methodology for developing multimedia and hypermedia software (MOOMH in Spanish *Metodología Orientada a Objetos para Desarrollar Software Multimedia e Hipermedia*) (Benigni, 2004);
- methodology for the development of educational multimedia software (MEDESME in Spanish *Metodología para el desarrollo de software multimedia educativo*) (García et al., 2016);
- thales methodology (Madueño cited by Dueñas et al., 2017);
- dynamic methodology for the development of educational software (Arias et al., 2015);

- educational software development (Quintero et al., 2005);
- model for the development of educational software based on competencies (MODESEC in Spanish *Modelo para el desarrollo de software educativo basado en competencias*) (Caro & Toscano, 2009).

Despite concerted efforts to attain learning objectives, the results are frequently needed to meet expectations. García et al. (2002) observe severe challenges persist in developing educational software despite the growing significance of computers and courseware in education. For example, in digital game-based learning, Zheng et al. (2024) highlight an ongoing unclear concerning students' learning performance. García et al. (2002) attribute these shortcomings in meeting learning expectations to an absence of theoretical and methodological foundation in the field and the complex integration of didactic elements with technological components and vice versa. This often leads to errors in strategy integration, conceptualization of the intended products, and communication among team members, among other issues.

Similarly, these problems continue to exist, affirming the relevance of this statement in the current context. Furthermore, Tzur et al. (2021) also underscores the inherent limitations of educational software when used in isolation. While the software can serve essential educational functions, it often needs more dynamic updates to keep pace with technological advancements, thus becoming quickly outdated. Moreover, the software's absence of real-time interaction and personalization limits its ability to adapt to individual learning needs and preferences, which is crucial for maintaining student engagement and motivation. As a result, even with advancements in educational technology, the effectiveness of such tools is compromised without the integration of human instruction and interactive elements that facilitate deeper understanding and retention of the learned material.

Thus, the research incorporates an analysis based on selected methodologies and methods that predominantly follow a constructivist approach, chosen because of their frequent use in technical articles within the field (Quintero et al., 2005; García et al., 2016; Caro et al., 2009; Serna et al., 2012; Martínez & Montero, 2016; Dueñas et al., 2017; Marcano & Benigni, 2014; among others). These methodologies are advantageous as they significantly enhance various aspects of the investigation.

Therefore, two criteria are determined for the comparative analysis. The first is the technological one, in which software development or software life cycle is considered, but it should not be confused with a methodology; Bentley (2008) defines the cycle as something that only happens from its conception to its obsolescence and that it is recommended to contemplate in the methodologies. After a bibliographic review (Sommerville, 2011; Kendall & Kendall, 2011; Braude & Bernstein, 2011; Bentley, 2008), the most relevant elements are considered, and five phases are established, which are described from the proposal of Kendall & Kendall (2011):

- analysis;
- design;
- developing;
- tests;
- maintenance.

The second, the educational one, is established in the teaching-learning process (T-LP) in the same way as in the first criterion; it is determined from a theoretical investigation of the concept (Carballo, 1978; Medina & Mata, 2009; Yánez, 2016), of thus, three phases are defined and specified based on what was mentioned by Carballo (1978):

- planning;
- execution;
- evaluation.

It should be noted that the names of the phases established in this research may or may not coincide with those in the literature. Still, they were defined in this way to begin standardizing terms, which are used as the basis for analyzing specialized methodologies in developing educational software.

Benito (2023) conducts an analysis, highlighting the integration—or lack thereof—of software development processes with teaching-learning processes in various educational software development methodologies. Notably, methodologies such as Educational Software Engineering (Galvis, 1992) and the Dynamic Methodology for Educational Software Development (Arias et al., 2015) are commended for their comprehensive coverage of the software development life cycle phases, from analysis to maintenance, which includes critical planning and evaluation phases of the teaching-learning process. This holistic approach ensures that pedagogical elements are integral at every stage, yielding software that is not only functional but also educationally effective. In contrast, methodologies such as ADITE (Pole, 2003) and MODESEC (Caro & Toscano, 2009) demonstrate significant gaps by omitting crucial stages like 'execution' within educational contexts. Such omissions can lead to software that fails to fully integrate educational objectives, potentially subordinating educational needs to technical constraints, thereby impairing the software's utility in academic settings. Additionally, while models like MEDESME (García et al., 2016) and Thales (Madueño cited by Dueñas et al., 2017) strive to encompass a broad spectrum of processes from both software development and pedagogical spheres, they struggle with synchronizing these aspects to enhance the coherence and functionality of the final product.

Furthermore, it is observed that none of the methodologies reviewed fully incorporate the 'execution' phase within their frameworks, reflecting a predominantly technological bias that often sidelines educational components. This oversight can lead to erroneous assumptions and the neglect of critical elements during the workflow. The analysis, conducted from the perspective of a multidisciplinary work team, underscores the necessity of explicitly detailing both the procedural components and their execution. Variations in conceptual understanding among team members, stemming from diverse disciplinary perspectives, can result in conflicting interpretations, which may lead to discord within the development team. The review also highlights areas where existing methodologies fall short and where this project can contribute to improvements. Consequently, this research aims to develop a methodology for creating educational software that effectively integrates these crucial elements, techniques, and instruments, ensuring that the final product is not only technically sound but also robust in its educational functionality (Benito, 2023).

3 METHODOLOGY

This work follows a methodology based on the type of applied research; Martínez (2019) establishes that its scope is not limited to documenting a fact in terms of knowledge expansion but instead seeks to influence reality, problematizing, analyzing, and generating a proposal. To modify the presented situation.

In the first instance, the evolution of technology and education is established through the study of state-of-the-art technology and education, and the problem of the conjunction of these to establish the ET is shown. The ambiguity in the terminology used so far is also shown. Then, the construction of educational software without a pedagogical foundation since the difficulty of replicating or eliminating both functional and non-functional factors in other academic projects has been identified.

In the second stay, the comparison and analysis of eight existing methodologies were carried out using the comparative method, defined as the procedure of systematic comparison of objects of study that, in general, is applied to arrive at empirical generalizations and to verify hypotheses (Nohlen, 2020); therefore, through documentary research, the phases that make up the teaching-learning process and the life cycle of pedagogical and technological software development are identified and established, in the same way, the products or deliverables of these stages, as well as their importance and impact within their respective procedure. Subsequently, a search was done for methodologies that impact a socio-temporal context like the one that permeates this work to compare the phases and products between them later.

Based on the previous, the pedagogical and technological elements used for the proposal are established, such as instructional design, list of requirements, didactic planning, UML (Unified Modeling Language), and validation of experts from both areas. In educational theories or standards of software development and, with it, the conceptualization of the proposal. Joint Methodology for Educational Applications (JMEA) is made up of five phases established from the software development life cycle, defined by three components: Description, which specifies the justification, characteristics, and general overview of the stage; procedure, where it is indicated how it works and how it is implemented; products, which are the results or deliverables (some of them mentioned at the beginning of the paragraph), seeking control, monitoring and evidence of what has been developed. These aspects define the context in detail for an optimal understanding of the work team. With this, factors that may affect the final objective of the educational system are not inferred.

4 RESULTS AND DISCUSSIONS

4.1 JOINT METHODOLOGY FOR EDUCATIONAL APPLICATIONS (JMEA)

The proposal aims for a working group with defined roles, responsibilities, profiles, and desired academic backgrounds. However, the conditions only allow some of the elements of the proposed work team to be specified. In that case, the software development must be carried out including at least one specialist in education and another in technology, in constant and close interaction during the process, to standardize conceptualizations of technology and education and vice versa; however, the hierarchy and positions must be covered and defined in their entirety from the beginning of the project and in a joint agreement with the available human resources.

Once the human resource has been defined, it is necessary to establish the workflow during the development of educational software; therefore, the following sections provide a detailed breakdown of the methodology, techniques, and instruments to be used in carrying out a project for educational purposes.

4.1.1 Abstraction

The project combines the software development life cycle, the Joint Application Design (JAD or Joint Application Design), and pedagogical elements, such as instructional design, didactic sequence, and educational theories. In this way, five stages are established, and a transverse phase will permeate each of these:

1. techno-pedagogical analysis: it is the fundamental stage of the methodology; in this, the pedagogical and technological bases of the software to be developed will be obtained, processed, and proposed, as well as the process planning, which includes the tools and resources to be used;
2. pedagogical design: based on what was obtained in the previous phase, visual aids will be developed to help experts in each area understand and propose optimal solutions that meet the needs;
3. implementation: the solution is coded using software development tools;
4. tests: the programmer, team members, experts, and end users must verify the software or system's correct operation and ensure that the requested characteristics are met;
5. functional and educational maintenance: periodic reviews should be scheduled to establish and correct faults, optimize the application after the product is delivered, and ensure adaptability to user needs and scalability if a relevant update is necessary.

Finally, the JAD is applied during all the phases described through meetings between the pedagogical and technological areas, in which development agreements will be established, seeking approvals for implementation. For the meetings, three aspects must be covered: (1) preparation that will result in the agenda, (2) understanding session with knowledge maps[1] as a product, and (3) a review of the points established in the minutes (Carmel et al., 1994; Balda & Vicenzi, 1997).

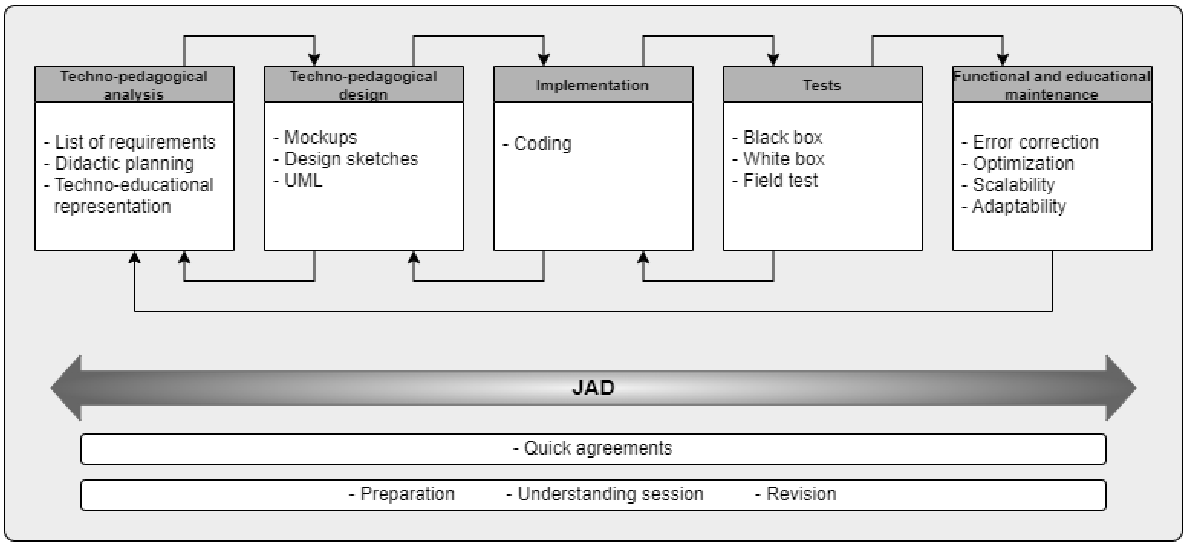
[1] Knowledge maps point the way to the sources of knowledge and information and structure the knowledge landscape with the representation of elements and interrelationships of the domains of knowledge. Like geographic maps, knowledge maps do not try to capture all aspects of knowledge but simplify and focus on some aspects of it (El Assafir et al., 2017).

The advantage that an education expert represents as part of the work team will be reflected in the quick agreements made now of doubt and added or discussed later in the formal sessions.

On the other hand, the advantage of having an education expert as part of the work team will be used since availability will facilitate quick agreements. In turn, the phases of JMEA have a sequential order; each stage cannot start without having completed its predecessor; however, it is possible to return if an omission or failure is identified during the execution of the current cycle in such a way that a cyclic flow is generated between contiguous phases; likewise, in carrying out functional and educational maintenance, it is necessary to return to the first stage, this to review and rethink the execution and optimization defects from the conception of the system, to reach a higher level in the product quality, because the developer group is forced to repeat the process from start to finish. This has been conceptualized in Figure 1.

Figure 1

The schematization of the Joint Methodology for Educational Applications.



Source: Benito (2023, p. 62).

Next, each JMEA phase is described in detail; likewise, the technological and pedagogical foundation used to conceive each element considered will be established.

4.2 PHASE 1: TECHNO-PEDAGOGICAL ANALYSIS

4.2.1 Description

Instructional design and software engineering are different; however, the literature on the pedagogical element is rediscovering concepts already treated mainly by the discipline of computational science; therefore, they are related in some way (Adnan & Ritzhaupt, 2018).

Specifically, the creation of teaching-learning environments has similarities with the life cycle of software development; for example, ADDIE, a generic instructional design model, encompasses five main activities: analysis, design, development, implementation, and evaluation; while the waterfall model, standard in the software industry, is made up of requirements analysis, design, development, implementation, verification, and maintenance, with the concordances between the two standing out (Gustafson & Branch, 2002; Willis, 2008; Adnan & Ritzhaupt, 2018).

In this sense, it is possible to infer several points where the homologations of concepts are viable or beneficial during educational software development. In this research, a close similarity between the analysis stages of requirements engineering and instructional design is proposed since both seek to obtain the requirements to be considered in the solution to be created and establish and understand the context in which it will be applied.

In this way, tools have been produced that play fundamental roles within their respective processes: a list of requirements to define the technological aspects, and didactic planning, with which the educational context will be determined; however, both instruments define particularities that are still difficult to integrate during the development of educational software, so it is proposed to use these elements in a way that brings together the areas involved and generates a techno-pedagogical definition.

4.2.2 Process

T-LP is a complex activity; however, if the variables encompassing the process can be defined optimally, the objectives are achieved with optimal results. Therefore, obtaining information to understand the educational context becomes crucial since the result is the foundation for all planning of how teaching and learning will be carried out; consequently, the formulation of a compilation instrument must be meticulous in obtaining a clear and concise pedagogical scenario that allows establishing the bases of technological development or the creation of computer resources. Therefore, it is assumed that technological development belongs to the educational context, so in this phase, techniques and tools are offered to carry out those, as mentioned earlier.

In the first place, interviews are established because it is one of the complete techniques to obtain information; a script is proposed that contains questions that seek to serve as a starting point during these; however, the experience of the interviewer will be a fundamental factor, since it will establish the route to follow, the complexity and the vocabulary to be used based on the interviewee, the inquiry and depth of details, among others. Likewise, the categorization does not seek to establish any separation or difference within the components of the system; its function is to define the objectives or the direction of the collected data. Concerning this, some questions obtain the same response, which is not a problem since they operate as control elements to determine the consistency of what is obtained.

In this context, based on an analysis of the pedagogical elements that should be considered in the T-LP (UNESCO, 1998; Pérez, 2021; Carballo, 1978; Medina & Mata, 2009; Yánez, 2016), added to the technical parameters of the international standard for Specification of Requirements IEEE830 (Gómez, 2011; Pfleeger, 2002), the aspects to be addressed in the interview are established: (1) the educational model, which determines the attitudes and aptitudes of the student sought to form; (2) pedagogical theory, which describes the roles of the actors in the T-LP and how to execute it; (3) teaching and learning strategies, which are the way of transmitting and acquiring knowledge; (4) the instructional design, which involves all the planning and execution processes of the T-LP; (5) technological development, which sets the technical course for the construction of computer components, which obtains the information that complements the previous ones and at the same time is enriched by them.

After the interview application, it is necessary to concentrate the information obtained and define what is required to be implemented so that at this stage, three products are requested, in which the techno-pedagogical bases that will serve as parameters during the construction of the study are achieved: Software.

4.2.3 Products

To start, it is sought that the educational context is determined, which will be achieved through the first four categories of the script to obtain techno-pedagogical requirements defined by the pedagogical expert. In this sense, the first product is didactic planning that establishes in a general way the variables to be considered, and that can affect the conception of educational software. These elements were established based on what was described by Digital Communities for Learning in Higher Education (2015), Murcia (2020), and Albarrán (2014).

In the same way, it is necessary to define the characteristics of the software from the technological point of view, given that the previously determined pedagogical base is not sufficient to establish what is going to be implemented; therefore, the second product of the phase will be a list of requirements that will contain the minimum elements to be considered during development, with which the functional and non-functional requirements of the system will be instituted.

Until now, it is glimpsed that the information concentrated and analyzed is observed from points of view that are not related; however, their similarity was already established at the beginning of this stage. In such circumstances, a link or bridge of integration of both perspectives is necessary; therefore, as a final product, a techno-educational representation is proposed, which seeks to combine what is defined in the didactic planning, and the list of requirements in this format software requirements are linked with its educational base. Therefore, the first application will be carried out under the second condition; likewise, it is essential to be as specific as possible in its completion since its result must be free from interpretations or design and implementation ambiguities.

4.3 PHASE 2: TECHNO-PEDAGOGICAL DESIGN

4.3.1 Description

At this point, communication bridges must be established through the technological and educational areas and the application of design techniques conventionally used throughout software development. Likewise, it is recommended to consider aspects of usability[2], conventions[3], and the determination or consideration of the learning channels of the end users, among others. Finally, it is sought that the complexity (in technical terms of software

[2] Extent to which certain users can use a product to achieve specific objectives with effectiveness, efficiency, and satisfaction in each context of use (International Organization for Standardization (ISO) cited by Dimuro, 2014).
[3] Conventions are a series of unwritten rules, acquired by everyday use, that everyone knows, understands, and interprets in practically the same way (Dimuro, 2014).

engineering) during the execution of the phase occurs gradually, starting with techniques understood by the entire work team and ending with merely engineering elements.

4.3.2 Process

In the first instance, the software *mockups will be developed* based on what was obtained in the previous stage, which is defined as a visual representation with a medium to a high level of fidelity of the appearance of the result and that shows basic concepts of its functionality (Cao et al., 2020). These must reflect the needs to be covered and be as specific as possible since the distribution of the elements of the system, the typography, the presentation of the information, the colors of the screen, and other components corresponding to the *front end* will be based on pedagogical aspects, so changing or substituting what is determined could affect the objective that is sought to be achieved.

Later, the design sketches will be created practically UML drafts; in these, the standard will not be followed, but it will be based on it; this assumes that UML has elements that are easy to understand regardless of the field of knowledge of the team members. It is essential to establish three rules: First, apply the symbology of the diagram as closely as possible to the UML standards; if it is not feasible, use characters whose meaning can be intuitively understood by the work group; second, use a conventional language avoiding technicalities that make it difficult to understand; likewise, the notes or sentences of each sketch will be as specific as possible; the third, both area experts must be involved in the preparation. The result will be the communication bridge between the educator and the programmer.

Finally, based on the design sketches, the UML diagrams will be elaborated; these must follow the standard without any modification since the programmer must understand them, and they will be the ones with which the coding will be developed.

4.3.3 Products

At the end of this phase, the deliverables will be the following:

- mockups;
- design sketches;
- UML diagrams.

4.4 PHASE 3: IMPLEMENTATION

4.4.1 Description

This stage refers to translating the design into a programming language, such as Java, Python, and C#. The implementation will be a purely engineering task, but the educational area will act as an auditor within its possibilities and capacities.

4.4.2 Process

The implementation will be done only when the design has been finished; the programmer will decide the optimal way in which he needs to develop the system; however, each decision should be discussed and validated with the education expert. Likewise, the coding must meet the following characteristics (Tsui et al., 2014; Gómez et al., 2019):

- readable code: it must be intelligible to other programmers;
- traceable code: there must be correspondence between each element of the code and every design component and vice versa so that these procedures can be traced;
- correct code: the encoding must satisfy the requirements;
- complete code: all requirements are met;
- good performance (performance): the system must work quickly and comply with the mentioned characteristics.

Finally, it is highly recommended that notations be established in the code and that this phase be documented using language that does not contain technicalities so that the work team can understand and thus make pertinent observations of the process.

4.4.3 Products

This stage's deliverable will be software with all the characteristics defined in the requirements; on the other hand, if you choose to develop the system by modules, these must be unified to comply with the phase's product. Likewise, a matrix that defines relevant software code is proposed to keep control during this point.

4.5 PHASE 4: TESTS

4.5.1 Description

In this stage, it is sought to establish the correct functioning of the system and that the established specifications have been met, and thus achieve the learning objectives, in the same way, take the impressions of the end users to generate the corresponding modifications if necessary.

4.5.2 Process

This phase is divided into a couple of moments, in the first two types of desktop tests will be executed: (1) white box, which checks that the internal units are implemented correctly, together with structures and relationships, to reduce internal errors; and (2) black box, which checks the external functions, as well as that the software contains the necessary specifications and meets the user's requirements (Rodríguez, 2012). The auditor will decide the conditions under which these techniques will be carried out if what is described is followed.

In the second moment, field tests will be carried out where end users (or those with very similar characteristics) will use the software for a limited period; later, the SUS (System Usability Scale) will be carried out. The SUS is a questionnaire with ten items that seek to determine or evaluate the level of usability of products and services, including computer systems (Brooke, 1996). It is worth mentioning that using SUS is only a proposal; however, other instruments can define a system's usability level.

4.5.3 Products

The evidence of desktop four and field tests will be considered to define the stage's fulfillment. For the first, a control format consisting of white and black boxes is established.

On the other hand, the evidence of the field tests will be carried out through the application of the SUS, remembering that this scale has its collection instrument; likewise, to consider that the project has an optimal level, SUS must give a rating between 80 and 100.

4.6 PHASE 5: FUNCTIONAL AND EDUCATIONAL MAINTENANCE

4.6.1 Description

Software development does not stop when the system is delivered but continues throughout its useful life; as soon as it is deployed in the educational context for which it was created (Sommerville, 2011), it is inevitable to continue adapting and scaling it so that it continues to meet the learning objectives.

4.6.2 Process

This stage is carried out after the delivery and deployment of the generated educational software; its duration will be the time agreed upon between the development team and the person in charge of requesting the system; the period is recommended to be at least one year. The phase will not follow an established order; only the following aspects will have to be considered:

- functional maintenance will investigate and correct the software's errors, seeking to optimize its operation. It will be carried out as soon as an end user reports a failure during the system's use; however, it must be scheduled periodically to continue with correct performance. These periods will be the product of agreements established between the client and the work team. Likewise, characteristics of the technical definition may be added or adapted;
- educational maintenance will be carried out at the beginning of each school year. The system will be adapted and scaled to cover the new learning needs based on the teaching-learning context represented by the current students. They will study the course or topic.

4.6.3 Products

Compliance with this phase will be measured by evidence of error correction, optimization, adaptability, or scalability. Given that the above defines a modification in the software, to carry out an adequate follow-up, a change control format is established with the following fields: Identifier, change to be made, description of the change, type of change, justification, and comments.

4.7 TRANSVERSAL PHASE: JAD (JOINT APPLICATION DESIGN)

4.7.1 Description

During the five stages, it is necessary to obtain, control, and implement the techno-pedagogical requirements clearly and concisely since it is common for there to be deficiencies in the way they will be represented in a computer system. To avoid this, JAD (Joint Application Design) is one of the leading methodologies for user involvement and participation in software development (Carmel et al., 1994).

JAD involves the user in a series of sessions that, traditionally, address the computer aspect in an unexplicit way; however, given the highly structured nature of JAD meetings and the essential role of the facilitator, they have the potential to become highly understandable meetings for the client (Carmel et al., 1994). Similarly, it is implemented on software development life cycles and structured analysis methodologies; pit reserves their organization while expanding user participation in the requirements specification and design stages (Balda & Vicenzi, 1997).

4.7.2 Process

JAD will be used throughout the entire software development process and must be applied constantly. In the first instance, the user's participation will be understood and partially replaced by the educational expert, who will act as a mediator and understander of the client's academic needs, remembering that the technical expert will conceive it from the section of his domain. In this role, the educational expert will constantly communicate with the client, validating the approvals made by the work team; likewise, the knowledge gathered from the problem to be solved will support you in proposing solutions that satisfy the requested requirements.

Similarly, communication between experts should be as continuous as possible, seeking to generate optimal agreements for technological and educational development; an essential point will be that they must agree with the established resolutions. This is done through meetings called understanding sessions, which will have the following three moments (Balda & Vicenzi, 1997):

- preparation: this phase consists of activities that prepare the next phase. The session leader (project manager) and the two analysts (educational expert and technological expert) participate;
- session: meetings where the system requirements and design are jointly developed; the session leader facilitates the group dynamics and guides the participants in preparing activities while the analysts record results;
- review: the products of the previous activities produce the standard outputs of the JAD; together, the session leader and the analysts translate the meeting's conclusions into a document. The design results are used to make a prototype.

In addition, as mentioned at the beginning of the chapter, the advantage of a multidisciplinary team is taken advantage of because the educational expert and the technical expert manage to make quick agreements (product of accessibility in the communication), which are similar to the agreements with the difference that they are not established in the understanding sessions but are taken when a modification is necessary, or some inconvenience arises in the implementation that requires an instant decision. In this way, situations that cannot wait for formal meetings are resolved; however, the decisions made during these will have to be discussed, and the advantages of the determination defined or if it is necessary to back down, in addition to rethinking a new solution, not forgetting to capture all this in a document.

4.7.3 Products

The expected results for each moment are the following:

- preparation: agenda;
- session: knowledge maps;
- review: minutes.

In this way, each element presented is considered relevant to control the development process better, so the omission or downplaying of any of them would establish situations that do little to favor the objective of educational software. The findings and deductions of the investigation are presented below.

4.8 VALIDATION

Two processes were established to validate the proposal: Validation by expert judgment and by work teams.

First, questionnaires were sent to 50 experts in educational technology from different nationalities, obtaining a collaboration of 30%, which is enough to perform an expert judgment (Cabero & Llorente, 2013). This percentage has developed educational software at different academic levels, with experiences covering various roles in the development of educational software.

The questionnaire responses showed that, in general, the specialists considered the JMEA proposal functional for educational software development. Aspects such as clarity, sufficiency, and adequacy of the proposed methods, techniques, strategies, and instruments were evaluated. Most of the specialists agreed or strongly agreed on these aspects.

Cronbach's Alpha was applied to measure the reliability and validity of the questionnaire, obtaining a value of 0.953, indicating excellent content and construct reliability. In addition, the specialists provided comments and suggestions, highlighting the importance of allocating adequate time to each stage and paying attention to the pedagogical approach. They also suggested including elements of accessibility, and adaptability, as well as fostering collaboration between knowledge areas.

Second, JMEA was presented to three educational software development teams to obtain feedback and validate its usability. Two evaluation moments were carried out: before the development process, improvements such as a glossary, filling instructions, and examples of systems created under JMEA were suggested; after development, the SUS questionnaire was used to measure perceived usability. The results showed a high acceptance of the methodology (82.5/100). However, the teams indicated the need to improve certain aspects, such as further specifying the questions to obtain requirements, working with experts, and creating manuals or instructions.

Each work team developed educational software; these systems are used in their corresponding teaching-learning processes. Although provisional results can each work team developed educational software used in their corresponding teaching-learning processes. Although provisional results can be determined, it is preferable to allow for a long-term process to conduct a relevant and meaningful analysis of these products' impact.

5 CONCLUSION

The proposal seeks to guarantee quality educational software that meets the learning objectives by establishing a development process that addresses both academic and technological perspectives, with the help of elements from these two areas: Instructional design, didactic planning, list of requirements, UML, among others; with which a pedagogical-technological balance is established that defines equal importance of both approaches.

During the comparative analysis, elements, products, and phases considered relevant for the software construction were established, which needed to be found clearly and in their entirety in the methodologies studied. For example, educational software engineering (Galvis, 1992) has been frequently cited due to the level of detail of each element, as well as for being a pioneer in its objective; however, some characteristics of the teaching-learning process, are omitted, such as execution, in the same way, a technical language of both areas is glimpsed, making it difficult to understand and implement it with multidisciplinary teams; additionally, like MEDESME (García et al., 2016) dynamic methodology for the development of educational software (Arias et al., 2015) and the other methodologies analyzed to have a technological approach, falling into the recurring problem of the development of educational software and, thereby occasionally not reaching the learning objective. JMEA seeks to highlight the importance of these components by integrating them explicitly and visibly for the work team during the development process, such as integrating planning and didactic sequence as one of the products of software development, among other situations.

This research has developed and validated a comprehensive methodology for creating educational software, which intricately integrates educational theories with technological advancements. The methodology, grounded in an academic and technological analysis, offers a structured approach that seeks to improve the pedagogical effectiveness of educational software, ensuring it meets both educational needs and technological standards.

Throughout the investigation, the methodologies reviewed highlighted a crucial need for a framework that not only addresses the technical aspects of software development but also the pedagogical components essential for effective learning environments. The proposed methodology, therefore, fills this gap by providing a balanced approach that emphasizes the importance of user-centered design, iterative development, and ongoing assessment.

For future research, it is recommended to explore the scalability of this methodology across different educational settings and cultural contexts to validate its effectiveness and adaptability further. Additionally, longitudinal studies could be conducted to assess the impact of the developed educational software on learning outcomes over time.

By aligning educational objectives with technological capabilities, this methodology paves the way for developing educational software that is both innovative and impactful, ultimately contributing to enhanced learning experiences and outcomes.

ACKNOWLEDGEMENT

To the Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT), for their financial support through CVU number 857798.

REFERENCES

Abud, M. (2009). MeISE: Metodología de Ingeniería de Software Educativo. *Revista Internacional de Educación en Ingeniería*, *2*(1), 1-9. Retrieved from https://cutt.ly/5j8NOoM.

Albarrán, M. (2014). *Diseño instruccional de objetos de aprendizaje. Una propuesta con base en la metodología del CATED-UNAM sobre: "importancia de las categorías y las variables en la investigación social*". (Bachelor's degree). Universidad Nacional Autónoma de México.

Adnan, N. & Ritzhaupt, A. (2018). Software Engineering Design Principles Applied to Instructional Design: What can we Learn from our Sister Discipline? *TechTrends*, *62*, 77–94. Retrieved from https://doi.org/10.1007/s11528-017-0238-5

Arias, M., López, A. & Honmy, R. (2015). Metodología Dinámica para el Desarrollo de Software Educativo. *Virtual Educa*. Retrieved from http://hdl.handle.net/20.500.12579/4325.

Balda, M. y Vicenzi, A. (1997). *Administrador de Flujo de Tareas y Documentos para la Especificación de Requerimientos*. Argentina: Universidad Nacional de la Plata. Retrieved from http://sedici.unlp.edu.ar/bitstream/handle/10915/2158/Documento_completo__.pdf?sequence=1&isAllowed=y

Benigni, G. (2004). Una metodología orientada a objetos para la producción de software multimedia. *Saber*, *16*(1), 26-32.

Benito, D. & Soberanes, A. (2023). Recurso Educativo Abierto para la enseñanza del presente simple en inglés: propuesta que integra componentes pedagógicos y tecnológicos en su desarrollo. *Diálogos sobre educación*, *26*. Retrieved from http://dialogossobreeducacion.cucsh.udg.mx/index.php/DSE/index

Benito, D. (2023). *Metodología para la implementación de software educativo basada en elementos tecnológicos y educativos centrados en el proceso enseñanza-aprendizaje*. (Ph.D. degree). Universidad Autónoma del Estado de México.

Bentley, W. (2008). *Análisis de sistemas. Diseño y métodos* (7th ed.) Mc Graw Hill.

Braude, J. & Bernstein, M. (2011*). Software engineering: Modern approaches*. Waveland Press.

Brooke, J. (1996). SUS: A "quick and dirty" usability scale. In P. Jordan. *Usability Evaluation in Industry* (pp. 189–194). Taylor and Francis. Retrieved from https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale

Cabero, J. (2007). Tecnología educativa: Su evolución histórica y su conceptualización. In J. Cabero. *Tecnología educativa* (pp. 13-28). McGraw-Hill. Retrieved from http://novella.mhhe.com/sites/dl/free/8448156137/471653/Capitulo_Muestra_Cabero_8448156137.pdf

Cabero, J. & Llorente, M. (2013). La aplicación del juicio de experto como técnica de evaluación de las tecnologías de la información y comunicación (TIC). *Revista de Tecnología de Información y Comunicación en Educación*, *7*(2). Retrieved from http://servicio.bc.uc.edu.ve/educacion/eduweb/v7n2/art01.pdf

Cao, J., Ellis, M. & Khachatryan, N. (2020). *The guide to mockups. Mockups, methods, and best practices*. Retrieved from https://www.pdf-archive.com/2016/10/14/the-guide-to-mockups/the-guide-to-mockups.pdf

Carballo, S. (1978). Fases del proceso Enseñanza-Aprendizaje. *Revista educación*, *2*(2), 47-57. Retrieved from https://revistas.ucr.ac.cr/index.php/educacion/article/view/18680

Carmel, E., George, J. & Nunamaker, J. (1995). Examining the process of electronic-JAD. *Journal of End User Computing*, *7*, 13-22. Retrieved from 10.4018/joeuc.1995010102

Caro, M. & Toscano, R. (2009). MODESEC: Modelo para el desarrollo de software educativo basado en competencias. *Nuevas Ideas En Informática Educativa*, *5*, 188-200. Retrieved from http://www.tise.cl/volumen5/TISE2009/Documento23.pdf.

Dueñas, D., Toscano, R., Gómez, A. & Caro, M. (2017). Sinopsis de metodologías y modelos de software educativo. *Acta ScientiÆ InformaticÆ*, *1*(1). 70-74.

Flores, I. (1997). *Una aplicación de fundamentos teóricos-conceptuales de los procesos de enseñanza-aprendizaje al diseño de un diplomado en computación para adultos* (Maestría). Universidad Autónoma de Nuevo León.

Galvis, A. (1992). *Ingeniería de software educativo* (1st ed.). Ediciones Uniandes.

García, E., González, J., Pérez, M., García, J. & Valdés, V. (2002). *¿Existe una situación de crisis del software educativo?* Memoria del VI Congreso Iberoamericano de Informática Educativa. Vigo, España: Universidad de Vigo. Retrieved from https://www.researchgate.net/publication/242579286_Existe_una_situacion_de_crisis_del_software_educativo

García, E., Vite, O., Navarrete, M., García, M. & Torres, V. (2016). Metodología para el desarrollo de software multimedia educativo MEDESME. *CPU-e. Revista de Investigación Educativa*, (23), 216-226.

Gimeno, J. (2008). *Comprender y transformar la enseñanza* (12th ed.). Ediciones Morata.

Gómez, M. (2011). *Notas del curso: Análisis de requerimientos*. México: Universidad Autónoma Metropolitana. Retrieved from http://ilitia.cua.uam.mx:8080/jspui/bitstream/123456789/168/1/619%20-%20G%c3%b3mez%20Fuentes%20Mar%c3%ada%20del%20Carmen.pdf

Gómez, M., Cervantes, J. & González, P. (2019). *Fundamentos de ingeniería de software*. Universidad Autónoma Metropolitana. Retrieved from http://www.cua.uam.mx/pdfs/conoce/libroselec/Fundamentos_Ing_SW-VF.pdf

Gustafson, K. & Branch, R. (2002). *Survey of instructional development models* (4th ed.). Syracuse: ERIC Clearinghouse on Information & Technology.

Hammad, R., Khan, Z., Safieddine, F., & Ahmed, A. (2020). A review of learning theories and models underpinning technology-enhanced learning artefacts. *World Journal of Science, Technology and Sustainable Development*, *17*(4), 341–354. Retrieved from https://doi.org/10.1108/WJSTSD-06-2020-0062

Heredia, Y. & Sánchez, A. (2013). *Teorías del aprendizaje en el contexto educativo* [Ebook] (1st ed.). Editorial Digital, Tecnológico de Monterrey. Retrieved from http://prod77ms.itesm.mx/podcast/EDTM/P231.pdf.

Hinostroza, E., Rehbein, L., Mellar, H., & Preston, C. (2000). Developing educational software: a professional tool perspective. *Education and Information Technologies*, 5(2), 103–117. Retrieved from https://doi.org/10.1023/A:1009699417462

Kendall, K. & Kendall, J. (2011). *Análisis y diseño de sistemas.* (8th ed.) Prentice Hall. Retrieved from http://cotana.informatica.edu.bo/downloads/ld-Analisis%20y%20Diseno%20de%20Sistemas_Kendall-8va.pdf

Knippers, J., Kropp, C., Menges, A., Sawodny, O., & Weiskopf, D. (2021). Integrative computational design and construction: Rethinking architecture digitally. *Civil Engineering Design*, *3*(4), 123–135. Retrieved from https://doi.org/10.1002/cend.202100027

Liviu, M. (2014). Comparative study on software development methodologies. *Database Systems Journal*, *5*(3), 37–55. https://cutt.ly/1j8NgyM.

Marcano, I. & Benigni, G (2014) Análisis de alternativas metodológicas para el desarrollo de software educativo. *SABER. Revista Multidisciplinaria del Consejo de Investigación de la Universidad de Oriente*, *26*(3), 297-304. Retrieved from https://www.redalyc.org/pdf/4277/427739473009.pdf

Medina, A. & Mata, F. (2009). *Didáctica general*. *UNED*. Pearson education. Retrieved from https://ceum-morelos.edu.mx/libros/didacticageneral.pdf

Mohebi, L. (2021). Theoretical Models of Integration of Interactive Learning Technologies into Teaching: A Systematic Literature Review. *International Journal of Learning, Teaching and Educational Research*, *20*(12), 232–254. Retrieved from https://doi.org/10.26803/ijlter.20.12.14

Munna, A., & Kalam, M. (2021). Teaching and learning process to enhance teaching effectiveness: a literature review. *International Journal of Humanities and Innovation (IJHI)*, *4*(1), 1–4. https://doi.org/10.33750/IJHI.V4I1.102

Murcia, M. (2020). *Diseño instruccional para profes: Guía para la innovación educativa con TIC* (1st ed.). Universidad Santo Tomás. Retrieved from https://repository.usta.edu.co/bitstream/handle/11634/22541/Obracompleta.Coleccionmodular.2020Murciamiguel.pdf?sequence=1

Nohlen, D. (2020). Capítulo tercero: El método comparativo. In *Antologías para el estudio y la enseñanza de la ciencia política. Volumen III: La metodología de la ciencia política* (pp. 41-57). Biblioteca Jurídica Virtual.

Pérez, H. (2021). Discurso general del encuadre pedagógico / Interviewed by D. Benito on 06/03/2022.

Pfleeger, S. (2002). *Ingeniería de software, teoría y práctica*. Pearson Education.

Pineda, L. (Coord.). (2017). *La Computación en México por especialidades académicas* (1st ed.). 167-194. Academia Mexicana de Computación, A. C.

Pole, M. (2003). Aproximación a un Modelo de Diseño: ADITE. *Docencia Universitaria*, *1*(4), 67-83.

Quintero, H., Portillo, L., Luque, R. & González, M. (2005). Desarrollo de software educativo: una propuesta metodológica. *TeloS*, *7*(3), 383-396. https://www.redalyc.org/articulo.oa?id=99318837004.

Ríos, P. (2001). Concepción del Software Educativo desde la perspectiva pedagógica. *Quaderns Digitals*, (24).

Rodríguez, E. (2012). Conceptos básicos de ingeniería de software. Presented at CINVESTAV-Tamaulipas. Retrieved from https://www.tamps.cinvestav.mx/~ertello/swe/sesion01.pdf

Schunck, D. (2012). *Teorías del aprendizaje* (6th ed.). Pearson Educación.

Serna, E., Castro, C. & Botero T. (2012). SEDLO: software engineering for developing learning objects. *EATIS '12: Proceedings of the 6th Euro American Conference on Telematics and Information Systems*. 2012, 347-353. Retrieved from https://dl.acm.org/doi/10.1145/2261605.2261658

Sommerville, I. (2011). *Software Engineering* (9th ed.). Pearson.

Tsui, F., Karam, O. & Bernal, B. (2014). *Essentials of Software Engineering* (3rd ed.). Jonas & Bartlett Learning books.

Tzur, S., Katz, A., & Davidovich, N. (2021). Learning supported by technology: Effectiveness with educational software. *European Journal of Educational Research*, 10(3), 1137-1156. Retrieved from https://doi.org/10.12973/eu-jer.10.3.1139

UNESCO. (1998). *Declaración Mundial sobre la Educación Superior en el siglo XXI: Visión y Acción*. Conferencia Mundial sobre la Educación Superior en el siglo XXI. Retrieved from https://unesdoc.unesco.org/ark:/48223/pf0000113878_spa

Urías, M., Torres, G., Valdés, A. & Antelo, M. (2015). Aportes y reflexiones sobre la educación mediada por tecnologías. In *Teorías que sustentan la Tecnología Educativa* (pp. 38-50). TABOOK. Retrieved from https://www.researchgate.net/publication/290447950_Teorias_que_sustentan_la_Tecnologia_Educativa

Willis, J. (2008). *Qualitative research methods in education and educational technology*. Charlotte: Information Age Publishing Inc.

Yánez, P. (2016). El proceso de aprendizaje: fases y elementos fundamentales. *Revista San Gregorio*, *11*, 70-81. Retrieved from https://www.researchgate.net/publication/313843119_El_proceso_de_aprendizaje_fases_y_elementos_fundamentales

Zheng, Y., Zhang, J., Li, Y., Wu, X., Ding, R., Luo, X., Liu, P. & Huang, J. (2024). Effects of digital game-based learning on students' digital etiquette literacy, learning motivations, and engagement', *Heliyon*, *10*(1). Retrieved from https://doi.org/10.1016/j.heliyon.2023.e23490

Žužek, T., Gosar, Ž., Kušar, J., & Berlec, T. (2020). Adopting Agile Project Management Practices in Non-Software SMEs: A Case Study of a Slovenian Medium-Sized Manufacturing Company. *Sustainability*, *12*(21), 9245. Retrieved from https://doi.org/10.3390/su12219245