Comparative Analysis of Multimedia Software Development Methodologies Under the ISO/IEC/IEEE 12207 Standard: A Process Mapping-Based Approach

Giannina Núñez Marín[(1)]
ORCID iD: 0000-0003-4436-3703
Correo electrónico: giannina.nunez@up.ac.pa
Diego Santimateo Gálvez[(1)]
ORCID iD: 0000-0002-1999-1743
Correo electrónico: diego.santimateo@up.ac.pa
Carmen C. Cortés Campos[(1)]
ORCID iD: 0000-0003-2269-0797
Correo electrónico: carmen.cortez@up.ac.pa
Yahaira Y. Juárez Ortega[(1)]
ORCID iD: 0000-0002-3973-6758
Correo electrónico: yahaira.juarez@up.ac.pa
[(1)] Facultad de Informática, Electrónica y Comunicación, Centro Regional Universitario de Veraguas – Universidad de Panamá

Abstract

The development of multimedia software entails distinctive characteristics regarding the integration of diverse data types, interactivity, and user experience—factors often insufficiently addressed by traditional software engineering methodologies. This study conducted a comparative analysis of the life cycle phases in eight multimedia software development methodologies in relation to the technical processes defined by the ISO/IEC/IEEE 12207 standard. The analysis identified applied software engineering methods, the application content domain, validation practices, and recommended documentation techniques. The research methodology involved data collection from various bibliographic sources, selection of relevant methodologies (1990–2025), and process mapping to align the life cycle phases of each methodology with the technical processes of the ISO/IEC/IEEE 12207 standard. Results revealed that the examined methodologies addressed between 50% and 75% of the standard's technical processes, leaving notable gaps in validation and documentation. One key limitation of the study lies in the absence of empirical validation through real-world projects, highlighting the need for future case studies and automated mapping tools. The findings offer scientific criteria for selecting and adapting multimedia software development methodologies, contributing to improved quality in both educational and commercial software products.

Keywords: Software life cycle, ISO/IEC/IEEE 12207, process mapping, software development methodology, multimedia software

Introduction

Multimedia software development involves particular considerations due to the types of data it manages and the functional characteristics of the software itself, both of which significantly influence user experience. Software functionality depends on the interaction between users and peripheral devices, meaning that application execution relies on user decisions made while interacting with various elements or characters within the environment.

Multimedia products fall into two categories: interactive and non-interactive. Within this context, the focus rests on interactive multimedia products—software applications that incorporate multimedia components such as event-driven applications, multimedia-based web applications, interactive multimedia learning materials, multimedia communication systems, and multimedia entertainment systems (Al-Jabari et al., 2019; Hannington & Reed, 2002).

In general, software functions as a system governed by a life cycle; thus, the stages from inception to retirement require definition within methodologies composed of development processes and activities. These, in turn, serve as frameworks for communication and understanding (ISO/IEC/IEEE, 2017). Software engineering entails the application of a systematic, disciplined, and quantifiable approach to software development, operation, and maintenance. This discipline encompasses processes, methods, and tools designed to support the creation of complex computer-based systems (ISO/IEC/IEEE, 2017; Pressman, 2010). Software development typically employs methodologies—understood as systems of practices, techniques, procedures, and rules—alongside methods, which consist of structured procedures for achieving deliverables throughout the software life cycle, and models, which function as abstractions to support software comprehension, design, and communication (Project Management Institute, 2021; Washizaki, 2024).

Within the multimedia software domain, specific methodologies address the phases of the software life cycle. These approaches aim to respond to the unique challenges involved in multimedia application development by considering aspects such as interactivity, usability, and user experience. In educational contexts focused on multimedia application development, students often encounter gaps regarding documentation practices and life cycle management of this type of software. As noted by Al-Jabari et al. (2019), students frequently struggle when analyzing, designing, and developing multimedia content for their projects, largely because no single, widely adopted methodology exists for multimedia product development. As a result, students tend to adopt traditional software engineering methodologies—such as waterfall, incremental, or agile models—that fail to adequately address the specific requirements of multimedia development. Many software process models obscure domain-specific characteristics through the use of generic terminology to describe generalized frameworks (Hannington & Reed, 2002).

The ISO/IEC/IEEE 12207:2017 standard on systems and software engineering provides a shared framework for software life cycle processes and establishes standardized terminology for the industry. This standard outlines the processes, activities, and tasks applicable to the acquisition, supply, development, operation, maintenance, and retirement of software systems, products, and services. However, the standard does not prescribe a particular life cycle model, development methodology, or specific technique, thereby allowing organizations the flexibility to choose and apply the approaches that best align with their specific needs (ISO/IEC/IEEE, 2017).

Nevertheless, the standard highlights several key elements that may relate to software development methodologies, such as:

- A common process framework for describing the software life cycle.
- A structured approach that integrates specialized disciplines and teams within a collaborative effort.
- Systematic definition of needs, documentation of requirements, system design, and validation within the development cycle.

This study aimed to carry out a comparative analysis of various multimedia software development methodologies in terms of the technical processes identified in the ISO/IEC/IEEE 12207 standard. Previous studies have explored general-purpose software development methodologies using different analytical criteria, such as the development approach applied (sequential, incremental, evolutionary, or agile) (Mujumdar et al., 2012); strengths and weaknesses related to requirement stability and project complexity (Despa, 2014; Gbaranwi et al., 2021; Islam & Ferworn, 2020; Saleh et al., 2017); and comparative studies between agile and waterfall methodologies, aimed at constructing decision trees to guide methodology selection (Mishra & Alzoubi, 2023). Tetteh (2024) analyzed agile software development models to identify their functionalities, strengths, and limitations, offering recommendations for industry decision-making. His study highlighted major challenges such as scalability, DevOps integration, data-driven decision-making, remote work adaptation, and continuous evolution of agile principles.

Marcano and Benigni (2014) evaluated methodologies for educational software development, focusing on factors such as life cycle model (waterfall, evolutionary, incremental), team composition, documentation requirements, and the inclusion of testing and validation stages. Parreira Júnior et al. (2009) examined the ISO/IEC/IEEE 12207 standard and concluded that it offers a roadmap for developing or acquiring high-quality software that meets user needs. However, compliance with all prescribed steps remains essential for ensuring software quality, making continuous project monitoring necessary. Irrazabal et al. (2011) investigated the relationship between agile practices—particularly SCRUM—and a subset of processes outlined in the ISO/IEC/IEEE 12207 standard. Their study, based on prior research and consulting with 25 companies, demonstrated the implementation of agile methodologies in line with the standard's outcomes. Notably, their analysis excluded the standard's technical processes, focusing instead on organizational facilitation and technical management processes.

To achieve this study's objective, the researchers conducted a process mapping exercise linking the technical processes from the ISO/IEC/IEEE 12207 standard to the life cycle processes of each selected methodology. The analysis also identified applied software engineering methods, the content domain of the targeted applications, validations conducted on each methodology, and the documentation techniques employed. This comprehensive analysis seeks to clarify how multimedia methodologies align with the software development standard and to assess the impact of these practices on the quality and effectiveness of both educational and commercial software. Consequently, learners and developers may apply scientific criteria to select the most appropriate multimedia development methodology.

Materials and Methods

This research adopts a qualitative, descriptive, and cross-sectional design. The applied methodology describes the data collection method, the analytical strategy for evaluating the selected methodologies, and a detailed account of both the ISO/IEC/IEEE 12207 standard processes and the software development methodologies under examination.

Data Collection Method

A literature review was conducted using databases such as Scopus, Emerald, Taylor & Francis, Google Scholar, and EBSCOHost. The search expression "Multimedia software development methodology" served to identify the target methodologies, focusing on publications from 1990 to 2025. Additional sources were retrieved using the expression "Comparative study of software development methodologies" to identify research aligned with the objectives of this study.

An initial search in Google Scholar using the phrase "Multimedia software development methodologies" yielded 93,500 results. A refined advanced search limited the timeframe to 1990–2025, reducing the number of results to 16,900. Further filtering by adding the phrase "Software life cycle" produced 3,870 results, which allowed for the identification of the most frequently referenced methodologies.

Analytical Strategy for Methodologies

The methodological analysis sought to answer the following research questions:

- Which technical processes from the ISO 12207 standard correspond to the life cycle stages of the studied methodologies?
- What software engineering models appear in the methodologies under review?
- What is the application domain (e.g., educational, commercial) targeted by each methodology?
- To what extent (percentage) do the selected methodologies align with the ISO/IEC/IEEE 12207 standard?

For each methodology, researchers identified the life cycle phases and performed a process mapping exercise to align those stages with the technical processes defined in the ISO/IEC/IEEE 12207 standard. A review of the existing literature revealed a lack of prior studies reporting on the mapping of multimedia software development methodologies to the standard's technical processes.

This analysis builds upon the methodological experience presented by Crisóstomo et al. (2017), who examined the relationship between two industry-relevant models: ISO/IEC/IEEE 12207 (software life cycle process standard) and CMMI-DEV (process improvement model). Their study confirmed that process mapping constitutes the most widely adopted comparison technique. As a systematic and structured tool, process mapping visually represents workflows, activities, interactions, and dependencies, thereby offering a comprehensive view of processes and clarifying the relationships among stages while uncovering opportunities for operational optimization. Pardo et al. (2012) similarly affirmed that mapping and comparison techniques prove useful for harmonizing models and standards such as CMMI, ISO 9001, ISO/IEC 15504, and ISO/IEC/IEEE 12207.

Drawing from Baldassarre et al. (2009) and Crisóstomo et al. (2017), this study measured the degree to which each methodology satisfies the technical processes of ISO/IEC/IEEE 12207. The

compliance percentage resulted from a comparative analysis between the life cycle stages of each methodology and the 14 technical processes defined in the standard.

Compliance percentage calculation followed these steps:

1. Process Mapping: Identification of the methodology's life cycle phases and their correspondence with the technical processes of ISO/IEC/IEEE 12207.
2. Coverage Scale: Assignment of a coverage percentage for each ISO process based on its presence and extent of implementation within the methodology:

    - Full Coverage (100%): All activities and tasks of the ISO/IEC/IEEE 12207 technical process appear fully implemented or explicitly addressed.
    - Partial coverage (50%): If only some aspects or tasks of each activity from all the technical processes of the ISO/IEC/IEEE 12207 standard were implemented or were taken into account.
    - Minimal coverage (25%): If the tasks of the activities in each technical process of the ISO/IEC/IEEE 12207 standard were indirectly considered or were mentioned.
    - No coverage (0%): If the tasks of the activities in each technical process of the ISO/IEC/IEEE 12207 standard were neither considered nor brought up.

To calculate the weighted average, the study summed the coverage percentages for all 14 processes and divided the total by 14:

$$\text{Average} = (\sum \text{coverage percentages} / 14) \times 100$$

For the qualitative analysis, researchers used Google's NotebookLM tool, applying the following prompt:

> "In the document by author ..., titled '...', a multimedia methodology consisting of multiple phases is described. Apply the process mapping criteria from 'criterioAsignacionPorcentaje2025' and determine the percentage to assign to each methodology phase based on its alignment with the ISO/IEC/IEEE 12207 technical processes. Present the results in a table that includes technical processes, the related methodology phases, and the assigned percentage."

NotebookLM received three reference documents: (1) the scientific article describing the methodology, (2) the document criterioAsignaciónPorcentaje2025.docx containing the defined coverage scale, and (3) the ISO/IEC/IEEE 12207 standard itself. Each researcher reviewed and interpreted the outputs independently.

NotebookLM, an AI-powered research assistant, generates responses exclusively based on the documents provided by the user, with the added advantage of identifying semantic patterns. Unlike traditional qualitative analysis tools—which typically require manual coding and categorization—NotebookLM streamlines pattern recognition and report generation. This capability proves essential in qualitative research, where traceability and fidelity to original data remain critical. While

NotebookLM served as a support tool in this study, the researchers retained responsibility for interpreting the data, assigning meaning, and articulating key findings.

ISO/IEC/IEEE 12207 Standard (ISO/IEC/IEEE, 2017)

ISO/IEC/IEEE 12207 is an international standard jointly developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). Officially titled "Systems and software engineering — Software life cycle processes", the standard was first published in 1995 and has undergone multiple revisions over the decades (ISO/IEC/IEEE, 2017).

This standard defines a comprehensive set of processes that span the entire software life cycle—from system conception and specification through operational deployment, maintenance, and eventual decommissioning. Three foundational principles underpin these life cycle processes: each process links directly to its products, activities, and tasks; processes operate independently from one another; and any process may execute individually within the life cycle. Each process includes a title, purpose, outcomes, activities, and tasks.

Table 1 presents the technical processes applied in the creation and use of a software system, whether as a model or an operational product. These technical processes remain applicable across all hierarchical levels of the software system architecture and throughout any stage of the life cycle.

Table 1. Technical Processes

Process	Purpose
Business or Mission Analysis	Define the problem, business opportunity, or mission; characterize the solution space; and determine potential classes of solutions capable of addressing the problem or exploiting the opportunity.
Stakeholder Needs and Requirements Definition	Define the requirements for a system capable of delivering the capabilities needed by users in a defined environment.
System/Software Requirements Definition	Translate the envisioned capabilities into a technical vision of a solution that satisfies users' operational needs.
Architecture Definition	Generate alternative system architectures, select those that satisfy system requirements, and represent them through a consistent set of views.
Design Definition	Provide sufficient detailed data and information about the system and its components to enable implementation consistent with the architectural entities defined in the system models and views.
System Analysis	Establish a rigorous information base to support technical understanding and decision-making throughout the life cycle. This process involves developing input data for technical evaluations and validating the usefulness and completeness of system requirements, architecture, and design.
Implementation	Realize a specific system element. This process transforms requirements, architecture, and design—including interfaces—into actions that produce

Process	Purpose
	a system element in accordance with the selected implementation technology.
Integration	Synthesize a set of system elements into a complete and functioning system (product or service) that fulfills system/software requirements, architecture, and design. This process includes assembling implemented elements and activating interfaces to enable the intended interoperability among system components.
Verification	Provide objective evidence that a system or system element meets its specified requirements and characteristics. This process identifies anomalies (errors, defects, or failures) in information elements, implemented system components, or life cycle processes by applying appropriate methods, techniques, standards, or rules.
Transition	Establish the capability for the system to deliver the services specified by stakeholder requirements in the operational environment.
Validation	Provide objective evidence that the system, when in use, fulfills its business or mission objectives and satisfies stakeholder requirements.
Operation	Employ the system to deliver its intended services. This includes establishing operational requirements, staffing the system, and monitoring service delivery and operator performance.
Maintenance	Preserve the system's service delivery capability. This process involves monitoring performance, recording incidents, applying corrective, adaptive, perfective, and preventive actions, and confirming restoration of full operational capacity.
Disposal	Terminate the use of a system or system element for a specific intended purpose, handle replaced or retired elements appropriately, and address critical disposal needs responsibly.

Source: Own elaboration.

Methodologies

Following an analysis of the methodologies retrieved from the search results, eight approaches were selected based on their alignment with the focus of this research—multimedia software development. These methodologies were chosen for their ability to define distinct development phases, thereby enabling a structured comparison with the technical processes established by the ISO/IEC/IEEE 12207 standard. The descriptions below present each methodology in ascending order of publication year.

Hypermedia Educational Applications Development Method (MDAEM)

This method integrates principles from software engineering and cognitive analysis to ensure both educational functionality and technical quality in multimedia applications. Its evolutionary life cycle enables continuous adjustments through iterative evaluations, thereby enhancing both pedagogical and technological effectiveness. The approach emphasizes final product quality and user

satisfaction, ensuring the application fulfills both educational and functional expectations (Valencia, 1998).

Methodology for Multimedia Product and System Development (MDPSM)

This methodology offers a structured yet flexible framework capable of adapting to technological evolution. It emphasizes project management, quality assurance, and risk management. Its primary goal involves improving the quality and efficiency of multimedia product design, production, and deployment by incorporating standards such as ISO 9001 to ensure consistency and process optimization. Activities are organized into three categories—development, management, and support—offering a comprehensive view of the product life cycle (Sherwood & Rout, 1998).

Object-Oriented Methodology for Multimedia and Hypermedia Software Development (MOOMH)

MOOMH focuses on developing multimedia and hypermedia systems using object-oriented techniques and software engineering principles. Its structure comprises three main phases: analysis, design, and implementation. Project management and control mechanisms form part of the framework, supported by a CASE tool that enhances design efficiency. The methodology aims to balance technical rigor with usability, thereby streamlining the development of multimedia projects for educational and informational purposes (Benigni, 2004).

Competency-Based Educational Software Development Model (MODESEC)

This model targets the creation of educational software through a competency-based approach, integrating pedagogical, learning, and software engineering principles. It structures the development process into five phases: instructional design, multimedia design, computational design, production, and deployment. The methodology ensures alignment between the software product and the specific competencies defined in the teaching-learning process (Caro Piñeres et al., 2009).

Methodology for the Development of Multimedia Learning Objects (MESOVA)

MESOVA concentrates on the development of virtual learning objects using a modular and evolutionary approach. The process begins with object conception, where pedagogical goals and requirements are defined. During the modular design and development phase, learning modules are created progressively. These modules are subsequently integrated and validated in a final environment. In the testing phase, the methodology assesses the object's impact on learning by comparing control and experimental groups. The final consolidation stage includes object deployment and documentation, followed by continuous evaluation for future improvement (Parra Castrillón, 2011).

Educational Multimedia Software Development Methodology (MEDESME)

This hybrid methodology combines multiple software development approaches—including spiral, incremental, waterfall, reuse-oriented, and evolutionary models—to ensure high-quality educational multimedia software. The process unfolds across eight phases: concept or pre-production, analysis of educational goals and technical requirements, interface and multimedia design, software

development, integration of multimedia elements, pedagogical impact evaluation and validation, final version production and distribution, and finally, the creation of supplementary materials such as user manuals and instructional resources. This structure supports a comprehensive approach to developing effective educational multimedia applications (García Sánchez et al., 2016).

Multimedia Project

Vaughan (2014, p. 1) defines a multimedia project as the software vehicle comprising messages and content displayed via computer, television screen, mobile device, or smartphone. Multimedia project development follows four major stages: planning and costing, design and production, testing, and delivery. These stages consist of specific phases such as analysis, pre-testing, prototype development, alpha development, beta development, and delivery. Throughout the process, feedback cycles incorporate testing and expert/user adjustments to refine the final product (Vaughan, 2014).

Multimedia Software Engineering Methodology (MISM)

Multimedia software development involves the convergence of two distinct domains: software engineering, which emphasizes the structured creation of digital products, and multimedia design, which focuses on visual and creative elements. Many traditional development methodologies fail to fully integrate multimedia production phases.

To bridge this gap, the authors propose an interdisciplinary methodology that combines the software development life cycle with the multimedia production process. This approach seeks to ensure product quality by efficiently integrating technical and creative aspects throughout the multimedia development project (Al-Jabari et al., 2019).

Results

In general, the life cycle stages of a software system encompass concept exploration, development, maintenance, and retirement, with transitions occurring between these stages. Each stage involves planning, execution, and evaluation activities. The sequence in which these stages unfold depends on the adopted life cycle model. Among the most commonly applied models are: sequential—such as the waterfall model—and non-sequential models—including incremental, spiral, iterative, and evolutionary approaches. Non-sequential models often incorporate agile techniques and methods (Washizaki, 2024).

It is worth noting that, according to SWEBOK V4.0 edited by Washizaki (2024), software engineering methods can be classified by their focus and application into the following categories: heuristic methods, formal methods, prototyping methods, and agile methods.

Table 2 identifies the life cycle stages and the corresponding life cycle model adopted by each of the studied methodologies—specifically, the sequential or non-sequential order in which life cycle

stages are applied in multimedia software development. Methodologies based on non-sequential models emphasize continuous validation and adaptability to multimedia environments.

Table 2. Stages and Life Cycle Model of the Methodologies Examined.

Metodology	Life Cycle Stages	Life Cycle Model
MDAEM (Valencia, 1998)	1. Problem description 2. Requirements definition and contract terms 3. Instructional design 4. Computational design 5. Production 6. Product evaluation	Evolutionary
MDPSM (Sherwood & Rout, 1998)	1. Initiation 2. Specifications 3. Design 4. Production 5. Review and Evaluation 6. Delivery and Implementation	Iterative, Prototyping
MOOMH (Benigni, 2004)	1. Analysis Model 2. Design Model 3. Implementation Model	Iterative and Incremental
MODESEC (Caro Piñeres et al., 2009)	1. Instructional Design 2. Multimedia Design 3. Computational Design 4. Production 5. Application	Overlapping Linear
MESOVA (Parra Castrillón, 2011)	1. Object Conception 2. Modular and Evolutionary Design and Development 3. Integration and Deployment 4. Learning Assessment 5. Consolidation 6. Continuous Evaluation and Improvement	Hibrid
Multimedia Project (Vaughan, 2014)	1. Planning and Costing 2. Design and Production 3. Testing 4. Delivery	Agile, Prototyping
MEDESME (García Sánchez et al., 2016)	1. Concept or Pre-production 2. Analysis 3. Design 4. Development 5. Implementation 6. Program Evaluation and Validation 7. Production	Iterative

Metodology	Life Cycle Stages	Life Cycle Model
	8. Creation of Supplementary Material	
MISM (Al-Jabari et al., 2019)	1. Requirements and Pre-production 2. Design and Production 3. Validation and Post-production 4. Evolution	Not specified

Source: Own elaboration

As shown in Table 2, all analyzed methodologies adopt a non-sequential life cycle model.

The life cycle stages listed in Table 2 indicate that half of the methodologies were specifically formulated for developing multimedia applications in the educational sector (Caro Piñeres et al., 2009; García Sánchez et al., 2016; Parra Castrillón, 2011; Valencia, 1998), while the other half apply to multimedia software development in general (Al-Jabari et al., 2019; Benigni, 2004; Sherwood & Rout, 1998; Vaughan, 2014). Further details regarding the main stages are presented in Table 5.

Based on the findings of Baldassarre et al. (2009) and Crisóstomo et al. (2017), the extent to which each methodology under study aligns with the technical processes of the ISO/IEC/IEEE 12207 standard has been determined. Table 3 summarizes the results of the process mapping between the ISO/IEC/IEEE 12207 technical processes and the selected software development methodologies. This mapping functions as a critical tool for comparing the standard with development methodologies by systematically aligning their process areas. Such alignment enables organizations to assess the compatibility and integration of methodologies with the standard, particularly in relation to the identified technical processes.

Table 3 reveals that the technical processes with the highest coverage include: Stakeholder Needs and Requirements Definition, Architecture Definition, Design Definition, and Implementation. Conversely, the processes with the lowest coverage—Operation, Maintenance, and Disposal—remain completely unaddressed by all methodologies. The Verification and Validation processes, which refer respectively to evaluation during development and the fulfillment of requirements when the system is in use, exhibit moderate levels of coverage across the methodologies.

Table 3

Coverage Percentage of ISO/IEC/IEEE 12207 Technical Processes by Studied Methodology

ISO/IEC/IEEE 12207 Technical Process	**METODOLOGY**															
	MDAEM (Valencia, 1998)		**MDPSM (Sherwood & Rout, 1998)**		**MOOMH (Benigni, 2004)**		**MODESEC (Caro Piñeres et al., 2009)**		**MESOVA (Parra Castrillón, 2011)**		**Multimedia Project (Vaughan, 2014)**		**MEDESME (García Sánchez et al., 2016)**		**MISM (Al-Jabari et al., 2019)**	
	FASE[a]	**P.C.[b]**	**FASE**	**P.C.**	**FASE**	**P.C.**	**FASE**	**P.C.**	**FASE**	**P.C.**	**FASE**	**P.C.**	**FASE**	**P.C.**	**FASE**	**P.C.**
Business or Mission Analysis	1	50%	1	50%	1	50%	1	50%	1	50%	1	50%	1	50%	1	50%
Stakeholder Needs and Requirements Definition	1, 2	50%	2	100%	1	50%	1, 3	50%	1	50%	1, 2, 3	50%	2	100%	1	50%
System/Software Requirements Definition	2	50%	2	100%	1	50%	1, 3	50%	1	50%	1	25%	2	100%	1	50%
Architecture Definition	3, 4	50%	3	100%	2	50%	3	50%	1	50%	2	50%	3	100%	2	50%
Design Definition	4	50%	3	100%	2	100%	2, 3	50%	1, 2	50%	2	50%	3	100%	2	50%
System Analysis	3	25%	3	50%	1	25%	3	50%	1	50%	1	50%	6	50%	1	50%
Implementation	5	50%	4	100%	3	50%	4	100%	1, 2	50%	2	50%	4	100%	2	50%
Integration	5	50%	4	100%	NA	0%	4	100%	3	50%	2	50%	NA	0%	2	50%
Verification	6	50%	5	50%	3	50%	4	50%	1, 2, 3	50%	3	50%	6	100%	NA	0%
Transition	6	50%	6	50%	3	50%	5	50%	3, 5	50%	4	50%	5	50%	NA	0%
Validation	NA	0%	5, 6	100%	3	25%	5	50%	1, 2, 3, 4	50%	1, 3	50%	6	100%	3	50%
Operation	NA	0%	6	25%	NA	25%	5	50%	5	25%	NA	0%	7,8	25%	NA	0%
Maintenance	6	25%	5, 6	25%	NA	0%	4	25%	5	50%	4	25%	NA	0%	4	50%
Disposal	NA	0%	NA	0%	NA	0%	NA	0%	NA	0%	NA	0%	NA	0%	NA	0%
Weighted Average	**35%**		**67%**		**37%**		**51%**		**44%**		**39%**		**62 %**		**35%**	

Note: a) The phase number corresponds to those listed in Table 2. b) P.C. refers to the percentage of coverage. c) NA indicates that no phase of the methodology addresses the technical process.

Source: Own elaboration

Table 4 summarizes the full or partial coverage of the technical processes across the various methodologies. Based on Tables 3 and 4, the methodology with the highest percentage of coverage is MDPSM by Sherwood & Rout (1998), followed by MEDESME by García Sánchez et al. (2016). The former focuses on the development of general-purpose multimedia products, while the latter targets the development of educational multimedia software.

Tabla 4 *Analysis of Technical Process Mapping*

Methodology	Percentage	Observation
MDAEM (Valencia, 1998)	35%	Covers 9 out of 14 technical processes partially. It does not address 3 of the 14 processes. This methodology was developed prior to the standard's consolidation.
MDPSM (Sherwood & Rout, 1998)	67%	Fully covers 7 out of 14 technical processes. Partially covers technical processes related to business analysis, system requirements, verification, and transition. Main weaknesses include operation, maintenance, and disposal.
MOOMH (Benigni, 2004)	37%	Fully covers only the design definition process. Partially covers 7 out of 14 technical processes.
MODESEC (Caro Piñeres et al., 2009)	51%	Fully covers 2 out of 14 processes; partially addresses 10 out of the 14 technical processes.
MESOVA (Parra Castrillón, 2011)	44%	Partially covers 12 out of 14 technical processes. Only minimally addresses the operation process.
Multimedia Project (Vaughan, 2014)	39%	Partially covers 10 out of 14 technical processes. Minimal attention given to requirements definition and maintenance.
MEDESME (García Sánchez et al., 2016)	62%	Fully covers 7 out of 14 technical processes. Three additional processes receive partial coverage.
MISM (Al-Jabari et al., 2019)	35%	Partially covers 10 out of 14 technical processes. The remaining four are not addressed.

Source: Own elaboration

Table 5 highlights elements that are specific to multimedia software development methodologies and are not explicitly present in more general software development approaches (e.g., XP, RUP, waterfall, spiral). These elements exhibit better coverage according to the process mapping analysis.

Table 5. *Analysis of Technical Processes Present Across All Studied Methodologies*

Metodology	Needs and Requirements Definition	Architecture Definition	Design Definition	Implementatio
MDAEM (Valencia, 1998)	Analysis of educational content and the pedagogical objectives of the software. Includes the identification of user needs and available resources.	Not explicitly stated, but inferred through the organization of content and its hypermedia structure.	Interface and hypermedia navigation design, with emphasis on human-computer interaction.	Software creation using hypermedia tools, including initial tests to verify functionality.
MDPSM (Sherwood & Rout, 1998)	Analysis of the target audience and their multimedia-related needs.	Navigation maps and storyboards. Identification of human activities supported by the interactive system.	Integration of interface design, navigation, and multimedia formats. Version control. Application of quality standards to ensure consistency and continuous improvement.	Strict control of changes in audiovisual and graphic media, not just in code. Multimedia integration testing to ensure compatibility among elements.
MOOMH (Benigni, 2004)	Analysis model in which system needs, project objectives, and involved stakeholders are identified.	System navigation graph.	Design model specifying each node of the graph and the associated multimedia elements.	Includes the technical development of the system based on previous models, using tools such as UML for web applications.
MODESEC (Caro Piñeres et al., 2009)	During the instructional design phase, educational factors are examined, and learning objectives and competencies to be developed are defined.	The multimedia phase addresses aesthetics and the communication system.	The computational design phase includes the development of software specifications and planning of technical implementation.	The production phase involves coding, integration, and validation of the developed software.
MESOVA (Parra Castrillón, 2011)	In the object conception phase, activities include characterization of the topic and educational level, as well as specification of functional and non-functional requirements	The design phase includes a graphical profile of the object, visualizing its constituent modules or subsystems.	In the design phase, global modular design and UML models for describing interactions and state transitions are linked to the software design.	The development phase includes coding and mapping necessary to build the learning object modules.
Multimedia Project (Vaughan, 2014)	Creative "look and feel" Content maps. Development schedule for multimedia elements	Navigation structure and system.	Storyboards Multimedia production (graphics, audio, video)	Authoring system
MEDESME (García Sánchez et al., 2016)	Identification of multimedia elements. Learning strategies and interactive tasks. User profile: educational level, prior knowledge,	Navigation design: navigation maps, storyboards, task analysis diagrams.	Instructional design based on pedagogical models. Integration of the graphical user interface with learning objectives.	Assembly of multimedia elements and their integration into the interface.

Metodology	Needs and Requirements Definition	Architecture Definition	Design Definition	Implementatio
	attitude toward technology. Learning environment.		Selection of tools for multimedia editing.	
MISM (Al-Jabari et al., 2019)	Study of the application domain	Part of component design; based on design theories and sets of software design rules.	Design and creation of multimedia components. Design and development of software components (architectural, interface).	Component integration

Source: Own elaboration

Another common element observed in the studied methodologies is the need for interdisciplinary teams to ensure project success. Among the required professionals are software developers, graphic designers, audio and video producers, instructional designers, and project managers.

In general, the proposed methodologies are grounded in literature reviews, drawing from existing software development approaches. The authors justify their proposals based on weaknesses identified in previously established methodologies. In most cases, the methodologies have been validated through practical application in the development of multimedia software products within university settings (see Table 6).

The methodologies by Sherwood & Rout (1998) and Al-Jabari et al. (2019), for instance, were reviewed and tested by expert peers involved in multimedia application development. Vaughan's (2014) proposal, in contrast, stems from his professional experience in multimedia production.

Table 6. Theoretical Foundation and Validation of the Studied Methodologies

Methodology	Foundation	Validation
MDAEM (Valencia, 1998)	Software engineering and ISO 9000-3 standard	Applied to the development of hypermedia educational applications at the Faculty of Engineering, University of Valle, Cali, Colombia.
MDPSM (Sherwood & Rout, 1998)	Literature review, influence of ISO/IEC 12207 and ISO/IEC 15504 standards	Peer review, testing with student groups in project development in the Multimedia Bachelor's program at Griffith University.
MOOMH (Benigni, 2004)	Literature review on object-oriented software engineering and educational software development methodologies	Applied for application development at the University of Oriente - Nueva Esparta campus.

Methodology	Foundation	Validation
MODESEC (Caro Piñeres et al., 2009)	Based on competency systems and educational software development processes	Applied in theses and coursework in the Educational Software Development area, led by the Edupmedia research group.
MESOVA (Parra Castrillón, 2011)	Literature review including elements of XP, RUP, UP, spiral, incremental and evolutionary life cycles, agile prototype development	Applied to the creation of 13 learning objects for computer programming at the Fundación Universitaria Católica del Norte.
Multimedia Project (Vaughan, 2014)	Based on project management	Applied in the multimedia software industry.
MEDESME (García Sánchez et al., 2016)	Literature review of software engineering	Applied in educational software production.
MISM (Al-Jabari et al., 2019)	Based on software engineering methodologies and multimedia production phases	Comparison and evaluation with ISO/IEC/IEEE 12207:2017 standard, expert peer review, application in student projects at Hebron University.

It is important to note that the SWEBOK V4.0, edited by Washizaki (2024), identifies two types of software modeling: structural modeling (physical or logical composition) and behavioral modeling (functions). To support this distinction, the Unified Modeling Language (UML) provides a rich set of diagrams.

In this regard, Table 7 summarizes the main techniques and tools recommended for documenting the multimedia software development process. Typically, the structural component of multimedia software is documented using navigation maps, interface sketches, and storyboards. In contrast, the behavior-oriented component—driven by events—is modeled using UML diagrams such as sequence diagrams and use case diagrams.

Furthermore, the tools and methods employed for development activities may vary depending on the type of product being created. For instance, in multimedia product development, idea generation and analysis often rely on storyboards, layouts, and sketches. In contrast, for general software products, idea generation and analysis typically involve interviews, observation, and use case descriptions.

Table 7 Software Modeling

Metodology	Suggested Documentation Techniques
MDAEM (Valencia, 1998)	Proposes the development of final documents for each phase, yet omits specific documentation techniques. Suggested documents include: instructional design with task analysis, standard format design for information units, and aesthetic specification of information items.
MDPSM (Sherwood & Rout, 1998)	Structures documentation according to each phase and the associated activities. Recommends documents such as the project plan, functional specifications, navigation maps, screen templates, storyboards and flowcharts, evaluation implementation strategies, and user manuals.
MOOMH (Benigni, 2004)	Involves requirement specification, navigation map development, lesson or information unit design, navigation graphs, sketches of information units, multimedia object libraries, and user manuals.
MODESEC (Caro Piñeres et al., 2009)	Suggests a competency design format, user manual, concept list, content design matrix, content diagrams, navigation maps, and interface descriptions. Also includes use case diagrams, class diagrams, object diagrams, sequence diagrams, and database models such as the Entity-Relationship Model (ERM) and Relational Model (RM).
MESOVA (Parra Castrillón, 2011)	Recommends an executive summary, user manual, and support plan. Advocates for concise, navigable formal documentation.
Multimedia Project (Vaughan, 2014)	Outlines key components for presenting the project proposal: needs analysis, target population description, creative strategy, and implementation model. Interactive multimedia design may adopt a storyboard represented through wireframing or schematic guidance. The structural design may be depicted using navigation diagrams (linear, hierarchical, non-linear, or composite).
MEDESME (García Sánchez et al., 2016)	Covers requirement and analysis documentation, general and technical psychopedagogical design files, technical specifications for software requirements, navigation schemes and maps, storyboards, task analysis diagrams, use case diagrams, entity-relationship diagrams, and interface template designs. Also generates technical manuals, user guides, and didactic activity manuals.
MISM (Al-Jabari et al., 2019)	Addresses feasibility studies, project planning, and risk analysis. During design and production phases, it recommends component modeling, architectural design, and interface design, without specifying particular documentation techniques.

Source: Author's own elaboration

Discussion

Classical software development methodologies, although widely adopted, fall short when addressing the specific challenges involved in developing multimedia applications—particularly within educational contexts, as highlighted by Al-Jabari et al. (2019). This finding reinforces the premise that multimedia applications demand differentiated methodological approaches, capable of managing the complexity inherent in the integration of multiple data types, supporting user-system interaction and user experience, and recognizing the role of multimedia software as a pedagogical aid. These conclusions align with Marcano and Benigni (2014), who emphasized the unique characteristics of educational software development.

Nonetheless, certain methodologies demonstrate applicability across both educational and commercial domains, underscoring their versatility and adaptability to diverse contexts.

Most of the methodologies examined adopt non-sequential life cycle models—such as iterative, evolutionary, agile, or hybrid approaches. This trend reflects a growing emphasis on flexibility, adaptability, and continuous feedback in multimedia software development, a finding consistent

with Tetteh (2024), who underscored the relevance of agile methodologies in the contemporary software industry.

The mapping between the phases of the methodologies and the technical processes outlined in ISO/IEC/IEEE 12207 revealed notable gaps and areas for improvement. In most cases, alignment with the standard remains partial, with significant deficiencies observed in requirements definition, design, implementation, and documentation processes. Notably, none of the methodologies examined address maintenance and disposal processes, despite their critical importance within the standard.

This omission may negatively affect the quality and sustainability of multimedia products, thereby emphasizing the need to adapt or design methodologies that explicitly integrate these normative processes. This outcome reinforces the argument advanced by Irrazabal et al. (2011), who stressed the importance of assessing the relationship between development methodologies and ISO/IEC/IEEE 12207—although their study focused on agile practices and technical management processes.

From a practical standpoint, the findings suggest that selecting or designing methodologies for multimedia development should incorporate practices that ensure compliance with essential technical processes, in accordance with international standards. This conclusion resonates with the insights of Parreira Júnior et al. (2009), who highlighted ISO/IEC/IEEE 12207 as a key framework for guiding high-quality software development.

The study relied on document analysis and theoretical process mapping, without direct empirical validation in real-world projects. This limitation restricts the generalizability of the results and highlights the need for future research that includes case studies and empirical validation in actual multimedia development environments.

Conclusions

Through the systematic mapping of the phases of the methodologies examined in relation to the technical processes defined by the ISO/IEC/IEEE 12207 standard, a technical gap emerges: all methodologies display limited coverage of critical standard processes such as operation, maintenance, verification, and validation. Only MDPSM, MODESEC, and MEDESME address over 50% of the technical processes required by ISO/IEC/IEEE 12207, followed by MESOVA with 44%, while the remaining methodologies fall below 40%. This pattern reflects an emphasis on pedagogical approaches, evidenced by broader coverage of processes related to needs definition, instructional design, and content implementation—while neglecting essential technical aspects.

The comparative analysis reaffirms the need to adapt or design methodologies specifically tailored for multimedia software. These approaches must not only address domain-specific features—such as interactivity, usability, and user experience—but also ensure compliance with international standards like ISO/IEC/IEEE 12207, in line with the recommendations of Parreira Júnior et al. (2009). To align with such standards, the integration of practices from ISO/IEC 15504 or CMMI-DEV (Crisóstomo et al., 2017) proves advisable.

Ultimately, this research documents existing multimedia software development methodologies, thereby responding to the needs identified by Al-Jabari et al. (2019). It lays the groundwork for

improving current methodologies and for designing more robust, context-sensitive alternatives aligned with international standards, thus contributing to enhanced quality and effectiveness in multimedia software development. The study also addresses the generalization problem highlighted by Hannington & Reed (2002).

Future research should focus on developing support tools for the automated mapping of processes in accordance with relevant standards—an approach consistent with Parreira Júnior et al. (2009)—as well as evaluating the impact of such methodologies on the quality of both educational and commercial multimedia products.

Referencias

Al-Jabari, M., Tamimi, T. K., & Ramadan, A.-A. N. (2019). Multimedia Software Engineering Methodology: A Systematic Discipline for Developing Integrated Multimedia and Software Products. *Software Engineering*, *8*(1), 1-10. https://doi.org/10.5923/j.se.20190801.01

Baldassarre, M. T., Piattini, M., Pino, F. J., & Visaggio, G. (2009). Comparing ISO/IEC 12207 and CMMI-DEV: Towards a mapping of ISO/IEC 15504-7. *2009 ICSE Workshop on Software Quality*, 59-64. https://doi.org/10.1109/WOSQ.2009.5071558

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

Caro Piñeres, M. F., Toscazo Miranda, R. E., Hernández Rozo, F. M., & David Lobo, M. E. (2009). Diseño de software educativo basado en competencias. *Ciencia e Ingeniería Neogranadina*, *19*(1), 71-98.

Crisóstomo, J., Melendez, K., Flores, L., & Dávila, A. (2017). Convergence Analysis of ISO/IEC 12207 and CMMI-DEV: Complementary Result from Systematic Literature Review. *CLEI Electronic Journal*, *20*(3). https://doi.org/10.19153/cleiej.20.3.7

Despa, M. L. (2014). Comparative study on software development methodologies. *Database Systems Journal*, *V*(3), 37-56.

García Sánchez, E., Vite Chávez, O., Navarrate Sánchez, M. Á., García Sánchez, M. Á., & Torres Cosío, V. (2016). Metodología para el desarrollo de software multimedia educativo MEDESME. *CPU-e: Revista de Investigación Educativa*, *23*, 216-226. https://doi.org/10.25009/cpue.v0i23.2169

Gbaranwi, Precious, B., Ojekudo, & Akpofure, N. (2021). A Comparative Analysis of Software Development Methodologies. *International Journal of Research and Innovation in Applied Science*, *06*(05), 159-166. https://doi.org/10.51584/IJRIAS.2021.6513

Hannington, A., & Reed, K. (2002). Towards a taxonomy for guiding multimedia application development. *Ninth Asia-Pacific Software Engineering Conference, 2002.*, 97-106. https://doi.org/10.1109/APSEC.2002.1182979

Irrazabal, E., Vásquez, F., Díaz, R., & Garzás, J. (2011). Applying ISO/IEC 12207:2008 with SCRUM and Agile Methods. En R. V. O'Connor, T. Rout, F. McCaffery, & A. Dorling (Eds.), *Software Process Improvement and Capability Determination* (Vol. 155, pp. 169-180). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-21233-8_15

Islam, A. K. M. Z., & Ferworn, Dr. A. (2020). A Comparison between Agile and Traditional Software Development Methodologies. *Global Journal of Computer Science and Technology*, *20*(2), 7-42. https://doi.org/10.34257/GJCSTCVOL20IS2PG7

ISO/IEC/IEEE. (2017). *Systems and Software engineering-Software life cycle processes* (ISO/IEC/IEEE 12207:2017(E)).

Marcano, I., & Benigni, G. (2014). Análisis de alternativas metodológicas para el desarrollo de software educativo. *Saber, Universidad de Oriente, Venezuela*, *26*(3), 297-304.

Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: A comparative analysis. *International Journal of System Assurance Engineering and Management*, *14*(4), 1504-1522. https://doi.org/10.1007/s13198-023-01958-5

Mujumdar, A., Masiwal, G., & Chawan, P. M. (2012). Analysis of various Software Process Models. *International Journal of Engineering Research and Applications (IJERA)*, *2*(3), 2015-2021.

Pardo, C., Pino, F. J., García, F., Piattini, M., & Baldassarre, M. T. (2012). An ontology for the harmonization of multiple standards and models. *Computer Standards & Interfaces*, *34*(1), 48-59. https://doi.org/10.1016/j.csi.2011.05.005

Parra Castrillón, E. (2011). Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje -MESOVA–. *Revista Virtual Universidad Católica del Norte*, *34*, 113-137.

Parreira Júnior, W. M., Ferreira Júnior, J. L. A., & Silva, L. P. da. (2009). UM ESTUDO DOS PROCESSOS DE CICLO DE VIDA DE SOFTWARE A PARTIR DA NORMA ISO 12207. *Intercursos Revista Científica*, *8*(2), 167-176.

Pressman, R. S. (2010). *Software engineering: A practitioner's approach* (7th ed). McGraw-Hill.

Project Management Institute. (2021). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Seventh Edition and The Standard for Project Management (ENGLISH)* (7th edition). Project Management Institute.

Saleh, S. M., Rahman, M. A., & Asgor, K. A. (2017). Comparative Study on the Software Methodologies for Effective Software Development. *International Journal of Scientific & Engineering Research*, *8*(4), 1018-1025.

Sherwood, C., & Rout, T. (1998). A Structured Methodology for Multimedia Product and Systems Development. *ASCILITE*, 617-625.

Tetteh, S. G. (2024). Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. *Asian Journal of Research in Computer Science*, *17*(5), 30-42. https://doi.org/10.9734/ajrcos/2024/v17i5436

Valencia, M. E. (1998). *Un método de desarrollo de aplicaciones educativas hipermedia*. Taller Internacional de Software Educativo TISE´97, Colombia.

Vaughan, T. (2014). *Multimedia: Making it work* (Ninth edition). McGraw-Hill Education.

Washizaki, H. (Ed.). (2024). *Guide to the Software Engineering Body of Knowledge v4.0: SWEBOK: Vol. v4.0*. IEEE Computer Society.