CAPÍTULO I

SUITE INTERACTIVAS EN AMBIENTES CON SUM

Pablo Emilio Ospina Rodríguez

Magister en Ingeniería Electrónica. Unipanamericana Fundación Universitaria Panamericana, Avenida (Calle) 32 No. 17-30, Teusaquillo, Bogotá, Colombia. Correo electrónico: peospinar@unipanamericana.edu.co. <https://orcid.org/0000-0002-1326-1184>.

Luis Guillermo Molero Suárez

Doctor en Ciencias Gerenciales. Unipanamericana Fundación Universitaria Panamericana, Avenida (Calle) 32 No. 17-30, Teusaquillo, Bogotá, Colombia. Correo electrónico: lmolero@unipanamericana.edu.co. <https://orcid.org/0000-0002-4117-8273>.

Judeira Batista De Abreu

Doctora en Ciencias Gerenciales. Universidad de La Guajira, Uniguajira. Km.5 Vía Maicao, Riohacha, La Guajira, Colombia. Correo electrónico: jjbatista@uniguajira.edu.co. <https://orcid.org/0000-0002-5058-9373>.

Claudia Patricia Ramírez Triana

Máster en Dirección Estratégica en Tecnologías de la Información. Fundación Universitaria del área Andina. Carrera 14A No. 70 A-34, Bogotá, Colombia. Correo electrónico: cramirez117@areandina.edu.co.

Jorge Alejandro Gaitán Albarracín

Ingeniero en Telecomunicaciones. Unipanamericana Fundación Universitaria Panamericana Avenida (Calle) 32 No. 17-30, Teusaquillo, Bogotá, Colombia. Correo electrónico: jalejandrogaitan@unipanamericana.edu.co.

Resumen

Este capítulo tuvo como propósito analizar la metodología SUM para el desarrollo de suite interactivas, aludiendo sus particularidades y ventajas en apoyo al desarrollo de esta tecnología. El estudio se ubicó en el paradigma cualitativo a través de un estudio documental, con diseño bibliográfico descriptivo. Para lograr lo anterior expuesto, las categorías analizadas fueron: 1) Inicios del desarrollo de software, 2) Metodologías de diseño, 3) Modelos de diseño, 4) Metodologías Ágiles, 5) Desarrollo de videojuegos, 6) Suites Interactivas, la integración tecnológica, 7) SUM, entornos de desarrollo y programación, 8) Diseño Algorítmico y Aplicaciones Modulares y 9) Competencias, creatividad e innovación en la programación en ambientes SUM. Como resultado, se realizó una revisión científica para determinar las ventajas de la metodología SUM en el desarrollo de suite interactivas basadas en videojuegos. Como conclusión se obtuvo, que la metodología SUM tiene un rol coherente en entornos de programación de videojuegos y muestra ser compatible con

las competencias y capacidades propias de los equipos de trabajo para este fin, dada la flexibilidad y asignación de responsabilidades y tareas descritas en su definición.

Palabras clave: metodología ágil, suite interactivas, SUM.

INTERACTIVE SUITE IN ENVIRONMENTS WITH SUM

Abstract

The purpose of this chapter was to analyze the SUM methodology for the development of interactive suites, alluding to its particularities and advantages in support of the development of this technology. The study was located in the qualitative paradigm through a documentary study, with descriptive bibliographic design. To achieve the above, the analyzed categories were: 1) Beginnings of software development, 2) Design methodologies, 3) Design models, 4) Agile methodologies, 5) Video game development, 6) Interactive Suites, technological integration, 7) SUM, development and programming environments, 8) Algorithmic Design and Modular Applications and 9) Competences, creativity and innovation in programming in SUM environments. As a result, a scientific review was carried out to determine the advantages of the SUM methodology in the development of interactive video game-based suites. In conclusion, it was obtained that the SUM methodology has a coherent role in video game programming environments and shows that it is compatible with the competencies and capacities of the work teams for this purpose, given the flexibility and assignment of responsibilities and tasks described in its definition.

Keywords: agile methodology, interactive suite, SUM.

Este capítulo es resultado de un proyecto de investigación intitolado “Desarrollo de Suite Interactiva con fines de rehabilitación para personas con movilidad reducida” que se encuentra ya finalizado.

Institución financiadora: Unipanamericana Fundación Universitaria Panamericana bajo el Código del Proyecto SIGP/Código entidad financiadora número PI242019.

Introducción

El desarrollo de suite interactivas basada en videojuegos, se ha convertido en una industria poderosa con tendencia a superar otras expresiones del entretenimiento como el cine o la música, esta expansión ha venido de la mano con la acelerada transformación digital tal y como lo expresa Molero, *et al.* (2019), pero sobre todo, gracias a la masificación del uso de herramientas intuitivas y los entornos de desarrollo que han dado lugar al surgimiento de numerosas empresas y programadores independientes que día a día generan gran cantidad de títulos nuevos con temáticas muy variadas (Hall *et al.*, 2020). Tal cantidad de nuevos productos, enriquecen el espectro posible de géneros como: juegos casuales, juegos para capacitación, rehabilitación, publicidad, juegos para la escena competitiva, entre otros.

Por otra parte, los sistemas gamificados son sin duda una expresión de cultura tecnológica propia de los nuevos tiempos donde las limitaciones técnicas ya no son una barrera para los desarrolladores, sean novatos o experimentados, independientes o grandes corporaciones productoras, pero pese a esta facilidad técnica existe un entretelón colmado de gran cantidad de cancelaciones, juegos con poca o nula visibilidad e incluso grandes decepciones para esta comunidad. No es raro hoy en día el despido masivo en este sector, jornadas extenuantes de desarrollo y otras tantas malas prácticas de la gestión en general, por lo cual es entonces necesario reforzar su gestión, es por ello, que el propósito de este capítulo persigue analizar la metodología SUM para el desarrollo de suite interactivas, aludiendo a sus particularidades y ventajas en apoyo al desarrollo de esta tecnología.

Para lógralo, se estableció el siguiente sistema de categorías donde figuran los constructos recogidos de la revisión científica: 1) Inicios del desarrollo de software, 2) Metodologías de diseño, 3) Modelos de diseño, 4) Metodologías Ágiles, 5) Desarrollo de videojuegos, 6) Suites Interactivas, la integración tecnológica, 7) SUM, entornos de desarrollo y programación, 8) Diseño Algorítmico y Aplicaciones Modulares y 9) Competencias, creatividad e innovación en la programación en ambientes SUM.

Fundamentación teórica

Inicios del desarrollo de software

Desde los inicios de la computación alrededor de 1940 (Booch, 2018), con la aparición de los primitivos computadores de válvulas de vacío, se inició la programación de software. En estas tempranas tecnologías, se introducía el programa directamente en la memoria del computador de forma manual, a pesar de ser programas relativamente cortos, era recurrente los errores tanto en la lógica del programa como en su inserción en el equipo, estos errores obligaban a los operarios a reinsertar todo el programa desde cero, retrasando la obtención de resultados y el inicio o ejecución de nuevos programas.

Con el paso del tiempo, la única seguridad para evitar errores en la lógica y digitación de un programa se limitaba a la experticia de los desarrolladores, haciendo de la programación más un arte que una ciencia exacta, esta limitación se manifestó como una incertidumbre constante en los tiempos de entrega y la calidad del trabajo obtenido, que llevó a que en muchas ocasiones no se encontraran los posibles errores en el software hasta después de su entrega al cliente.

Asimismo, con el aumento de la complejidad y el tamaño de los programas creados, este problema se acentuó hasta llegar a la crisis del software de 1968, tratado como un elemento sensible en la primera conferencia de La Organización del Tratado del Atlántico Norte (OTAN) de ese mismo año. Como solución a esta crisis, surgió la ingeniería de software como lo menciona (Mall, 2018) que consideró la programación como una ciencia cuyo quehacer debía formalizarse y hacerse de manera ordenada y planificada, esto se hace notorio, con la llegada de la arquitectura de software, mediante la cual se da estructura a un programa dividiendo este en diferentes elementos y modelando a partir de sus interacciones su organización.

A lo anterior, la norma ISO/IEC/IEEE (2011) señala que los elementos de un programa corresponden a: su abstracción en forma de entidades correlacionadas, sus interacciones, el ambiente de desarrollo, las directivas que orientan su diseño, su evolución y finalmente, su mantenimiento. Es por tanto, que la arquitectura de software da una visión holística del sistema, omitiendo los pormenores y la algoritmia del software a desarrollar, considerándolo en muchos escenarios como un ciclo de vida.

Metodologías de diseño

Dado que la arquitectura de software concibe sistemas de software orientados a la satisfacción de las necesidades particulares de un proyecto o cliente (McConnell, 2004), es necesario delimitar los requerimientos del sistema antes de establecer los bloques funcionales que lo conforman y la estructura de su interacción. Por ello, el primer paso, consiste en el levantamiento de requerimientos (Kendall y Kendall, 2005), lo cual puede y debe contemplar más allá de la funcionalidad del software, teniendo en cuenta, atributos de calidad, diseño y puesta en marcha (evolución en el tiempo), actualización, seguridad, reusabilidad, procesos de auditoria y ajuste, entre otros.

Desde otro punto de vista y enmarcado en el tema que se trata, el modelado de sistemas dentro de la arquitectura de software se analiza a la luz de diferentes vistas que obedecen a un enfoque de desarrollo, de esta forma, las vistas arquitectónicas profundizan en aspectos específicos del modelo a generar, facilitando así la caracterización de las necesidades del proyecto, facilitando la orientación de este en los aspectos relevantes para los diversos participantes de este.

Modelos de diseño

Estudios más constituidos sobre la arquitectura de software, han dado como resultado estructuras arquitectónicas formales contentivas de múltiples vistas resumidas que abordan tópicos estándar y los interrelacionan entre sí, donde las más empleadas se aprovechan para la construcción de modelos de trabajo.

Tabla 1
Arquitectura Empresarial Zachman

	Datos Qué	Procesos Cómo	Redes Dónde	Personas Quiénes	Tiempo Cuándo	Motivo Por qué	
Ámbito (Entorno)	Lista de intenciones del negocio	Lista de procesos hacia afuera	Lista de ámbitos del negocio	Lista de Interesados	Lista de ciclo del negocio	Lista de metas del negocio	Gerencia
Modelo Empresa (Conceptual)	Modelo semántico del negocio	Modelo de procesos de negocio	Modelo de interacción	Modelo perfil-rol	Plan maestro	Plan estratégico de negocio	Dueño
Modelo de sistemas de información (Lógico)	Modelo lógico de datos	Arquitectura del sistema de información	Arquitectura distribuida del sistema de información	Arquitectura de interfaces humanas	Estructura de procesamiento	Modelo de reglas de negocio	Diseñador
Modelo Tecnológico (Físico)	Modelo físico de datos	Diseño de aplicaciones	Arquitectura tecnológica	Aequitectura de presentación	Estructura de control	Diseño de reglas de negocio	Constructor
Especificaciones de detalle (Producto)	Especificaión de datos	Programas	Arquitectura de redes	Arquitectura de seguridad	Definición de tiempos	Especificación de reglas de negocio	Sub contratista
Empresa funcionando	Datos	Procesos	Redes	Organización	Programa	Estrategia	Empresa funcionando

Fuente: elaboración propia (2020).

A continuación, en la siguiente tabla se ubican las estructuras arquitectónicas formales más utilizadas:

Tabla 2
Estructuras Arquitectónicas Formales.

Arquitectura/Modelo	Descripción
Arquitectura Empresarial de John Zachman	Inicia con las preguntas descriptivas “qué”, “cómo”, “cuándo”, “quién”, “dónde” y “por qué” (tabla 1). Con el fin de mantener un enfoque orientado a cada grupo participante dentro del proyecto, se realiza este mismo planteamiento a partir de las preguntas descriptivas ya mencionadas, desde las perspectivas: contextual, conceptual, lógica, física y detallada, dando lugar así a un framework de trabajo fácilmente modelable como una matriz. A pesar de no centrarse como tal en una metodología de diseño, sí establece los elementos y correlaciones necesarias para culminar exitosamente el desarrollo de una solución software. (Zachman, 1987).

Tabla 2
Estructuras Arquitectónicas Formales. (Continuación)

Arquitectura/Modelo	Descripción
Modelo de Kruchten (Vistas 4+1)	<p>Desglosa los elementos del proyecto en cuatro vistas, a saber: Lógica: útil para el diseño orientado a objetos. Procesos: identifica los estadios del desarrollo y su orden de ejecución. Física: aborda la distribución del software en múltiples computadores, cliente y servidor. Desarrollo: se orienta al ambiente de desarrollo y la organización de los elementos del software a su interior. A su vez estas cuatro visiones tan disímiles del proyecto, se articulan forjando su interacción de casos de uso o escenarios de aplicación, aportando una cohesión global de todos los aspectos del software, de ahí su nombre de 4+1 (figura 1) siendo los escenarios el punto en común de las cuatro vistas. (Kruchten, 1995), (ISO/IEC/IEEE 42010, 2011).</p>
Modelo de Grady Booch (OBT)	<p>Esta metodología se subdivide en dos instancias itinerantes: Fase de análisis: previo al diseño de la solución, es necesario realizar el levantamiento de información y dar lugar a una primera estructura sobre la cual iniciar el desarrollo del sistema. Fase de diseño: en esta fase se reconocen cuatro estadios de desarrollo a ejecutarse de forma cíclica perfeccionando en cada iteración su desempeño, a saber: diseño lógico, diseño físico, prototipado y prueba. (Booch, 1993).</p>
Modelo de James Rumbaugh (OMT)	<p>Conocido como metodología OMT, se define como un sistema de cuatro pasos que contempla un análisis bastante pormenorizado e intensivo de las necesidades del sistema a diseñar, por lo que precisamente su fase de diseño suele ser bastante dinámica e intuitiva. Los pasos para su desarrollo son: análisis de objetos, diseño del sistema, diseño de objetos e implementación. (Rumbaugh, 1996).</p>
Modelo de Ivar Jacobson (OOSE)	<p>Concentra sus esfuerzos en el nivel informático, usuarios y negocio del sistema a desarrollar, construyendo para esto una suerte de narrativa en torno a los casos de uso e interacción, es decir, estableciendo una metodología para sacar el máximo provecho de los casos de uso que se puedan construir acerca de las necesidades del sistema. Esta metodología reconoce cinco técnicas fundamentales para el modelado de un sistema a partir de sus casos de uso: modelo de requerimientos, modelo de análisis, modelo de diseño, modelo de implementación y modelo de pruebas. (Jacobson, et al., 1993).</p>

Fuente: elaboración propia (2019).

Figura 1

Modelo “4+1” vistas de Kruchten

Fuente: elaboración propia (2019).

Como se puede observar, estas metodologías de las cuales hay múltiples variaciones según la necesidad del equipo, el proyecto y el cliente, poseen en general etapas o vistas bastante similares entre sí, y en cada una de estas instancias es posible emplear métodos descriptivos y de documentación propios, sin embargo en un intento por unificar las diferentes estructuras de descripción y diseño, fue desarrollado el lenguaje de modelado unificado UML (figura 2) (Rumbaugh *et al.*, 2000), (ISO/IEC 19501, 2005), el cual busca mediante un lenguaje estandarizado facilitar la descripción coherente de múltiples características del proyecto. Dicho lenguaje de modelado busca la descripción de un plano en el cual se mezclan de forma ordenada caracterizaciones conceptuales del sistema, permitiendo generar esquemas, diagramas y llevar a cabo la documentación detallada de los pasos de desarrollo de un proyecto. Cabe resaltar que UML no es como tal una metodología de desarrollo o un lenguaje de programación, se trata más bien de una serie de normas que establecen de manera formal, estándares para la representación de múltiples facetas de la identificación, diseño y documentación del proyecto.

Figura 2
UML y tipos.

Fuente: elaboración propia (2019).

Sin embargo, y pese a contemplar normas y estándares para muchas actividades de documentación en concreto, este no es el único estándar de amplia aplicación, existen otras opciones comunes como: diagramas de grafos, redes de Petri, metamodelo, entre otros.

Metodologías Ágiles

Las metodologías tradicionales, son de gran utilidad en proyectos pequeños con un desarrollo predecible, solo basta con realizar un correcto levantamiento de los requerimientos iniciales (Kendall y Kendall, 2005) para realizar un desarrollo exitoso y sin contratiempos en un corto periodo de tiempo, sin embargo, en los entornos de desarrollo actuales, donde existen proyectos de gran volumen, con requerimientos cambiantes y con equipos multidisciplinares que atiendan a las necesidades del contexto, surgen las llamadas metodologías ágiles (figura 3, metodología ágil SCRUM) (Fowler *et al.*, 1999).

Figura 3

Scrum

Scrum Process

Enter your subhead line here

Fuente: <https://winred.es/management/metodologia-scrum-que-es/gmx-niv116-con24594.htm>

El objetivo principal de las metodologías ágiles son la flexibilidad y adaptabilidad del desarrollo a los requerimientos en constante evolución (Kniberg, 2007), por lo que desde su definición, las metodologías ágiles (figura 4, metodología ágil Kanban) no se consideran como tal un proceso de desarrollo, se trata más de un marco de referencia para el desarrollo de proyectos (Kniberg y Skarin, 2010), esto implica, la definición de diversas reglas que deben ser simples y claras desde un inicio pero con un continuo proceso de revisión y mejora a fin de adaptarse de manera perpetua al flujo del proyecto, involucrando activamente al cliente entro del proceso.

Figura 4

Metodología Kanban

Fuente: elaboración propia (2019).

Como se mencionó, las metodologías ágiles (figura 5, Metodología Extreme Programming o XP) buscan orientar el proceso de desarrollo, facilitando su evolución y

adaptación a los cambios en los requerimientos iniciales, por lo cual, lejos de ser una camisa de fuerza se trata de un sistema reiterativo que realiza múltiples pruebas y ajustes dentro de su ciclo de vida considerando cuatro pilares fundamentales (Cohn, 2004) a saber:

- Dar espacio prioritario a la interacción del equipo por encima de los procesos o herramientas
- Centrarse en la funcionalidad y calidad del software a desarrollar, por sobre la documentación del proceso.
- Preocuparse por involucrar al cliente en el proceso de mejora continua, en vez de centrarse en el cumplimiento de obligaciones contractuales.
- Mantener el dinamismo en la toma de decisiones en pro de la mejora continua, en vez de limitarse a una serie de pasos a cumplir.

Figura 5
Metodología Extreme Programming o XP

Fuente: <https://lorbada.com/blog/2017/02/10/diferentes-metodologias-agiles/>

En ese orden de ideas, una de las metodologías ágiles de mayor aceptación es SCRUM. Más que una metodología se reconoce como un conjunto de buenas prácticas que buscan optimizar el flujo de diseño e implementación, así como impactar positivamente en la cultura organizacional del equipo de trabajo responsable del proyecto.

Esta metodología de forma similar a otras corrientes administrativas como Kaizen e incluso el mismo Toyotismo, segmenta el proyecto en pequeñas fases a desarrollar, en cortos periodos de tiempo, limitando la influencia de cada pequeña tarea particular al esfuerzo de unos pocos miembros del equipo, en consecuencia, se busca generar una alta concentración de esfuerzo en alcanzar pequeños entregables, priorizando sus tareas a fin de obtener el máximo resultado posible.

Desarrollo de suites interactivas basadas en videojuegos

Las suite interactivas basadas en videojuegos como medio interactivo y multimedia ha recorrido un largo camino desde sus inicios, enriqueciéndose constantemente de cara al usuario, con la inclusión de nuevos enfoques técnicos y narrativos, incrementando constantemente su capacidad de inmersión y su facultad única, comparado con otros medios como el cine o la literatura, de permitir al usuario tener un rol participativo en su desarrollo narrativo, tal como lo menciona Kuo *et al.*, (2017).

Entre otras cosas, el videojuego es un punto de confluencia de múltiples artes y disciplinas (Molero *et al.*, 2019) que pasa por la escritura de guiones o historia, la musicalización, el dibujo, la animación, la programación que implementa el sistema de físicas hasta elementos que son propias de los sistemas de software como las interfaces de usuario, la usabilidad, la accesibilidad y el soporte para interfaces de interacción hombre-máquina como es el caso del Kinect y los sistemas HMD (Head Mounted Display). Es normal entonces, que la integración e interacción de tal cantidad de elementos disímiles, manejados por equipos de trabajo interdisciplinarios, requiera zanjar diferencias entre los múltiples enfoques como jugabilidad, curva de aprendizaje o diversión, que emergen de las dinámicas propias del videojuego, como por ejemplo, la cantidad de vistas posibles en el desarrollo de un videojuego, donde (Schell, 2019) sintetiza más de 100 vistas diferentes de cara a esta implementación.

Es entonces, cuando poco a poco esta tecnología de desarrollo de suite interactivas basada en videojuegos, se ha convertido en una industria de alto impacto económico, al punto, que en gran cantidad de países su facturación es superior al de otras industrias del entretenimiento como el cine (Benito, 2006). Es por ello, que este mercado emergente debe en parte su rápido crecimiento a múltiples factores, tales como:

- El fortalecimiento de grandes empresas de desarrollo.
- Los grandes lanzamientos de videojuegos AAA y sus entregas anuales.
- Su importancia en la cultura popular.
- La remasterización de clásicos con grandes ventas.
- La democratización del desarrollo de videojuegos, producida por la masificación de las herramientas integrales para su desarrollo.
- El uso de los teléfonos inteligentes.
- Tendencias emergentes de los videojuegos.

Desde otra perspectiva, la inclusión de lúdica en ambientes como la salud, rehabilitación, entrenamiento, educación, mercadeo o la publicidad no son para nada nuevas, desde la década de los setentas (Wilkinson, 2016) surgieron términos como “Juegos serios” que aluden un diseño en apoyo a procesos concretos. En adición a lo anterior expuesto, resulta asimismo interesante resaltar el carácter inmersivo de los videojuegos como estrategia para

el desarrollo de actividades tales como: potenciar terapias de rehabilitación o entrenamiento físico, campañas publicitarias, estrategias de responsabilidad social, sensibilización e instruir en temas complejos añadiendo un componente lúdico al objeto de estudio.

A manera de conclusión, las tendencias emergentes que ofrecen las suite interactivas basadas en videojuegos cuyo fin ulterior es el entretenimiento, la capacitación, el entrenamiento, la enseñanza, entre otros, exigen los mismos requerimientos y condiciones que el desarrollo de sistemas gamificados convencionales, de tal forma que autores como López (2016) observan resultados equiparables al emplear un videojuego diseñado específicamente como herramienta para impartir algún conocimiento, que al uso de videojuegos comerciales, que bajo un contexto adecuado y con las indicaciones correctas, pueden acercar al estudiante al mismo objeto de aprendizaje. En general, el empleo de esta estrategia para la construcción del conocimiento, es decir, la capacidad de un software lúdico para instruir sobre algún tema, recae en gran parte en su usabilidad y experiencia de usuario (UX) (Molero *et al.*, 2019), siendo posible adaptarlo a cualquier contexto, como la visión reportada por López (2016).

Metodología

La metodología del presente capítulo se desarrolló bajo el paradigma cualitativo donde en este particular Hernández *et al.*, (2010) expresan, “la investigación cualitativa se enfoca en comprender los fenómenos, explorándolos desde la perspectiva de los participantes en un ambiente natural y en relación con su contexto”. p. 364, asimismo, el tipo de investigación es documental, ya que se pretende ampliar y profundizar el conocimiento a través de fuentes bibliográficas y documentales.

En este particular, Nava (2002) afirma que la investigación documental, es una investigación formal, teórica abstracta si se quiere, por cuanto recoge, registra, analiza e interpreta la información contenida en documentos en soporte de información registradas, es decir, constituye un proceso de búsqueda, selección, lectura, registro, organización, descripción, análisis de interpretación de los datos extraídos de fuentes documentales existentes, en torno a una problemática, con el fin de encontrar respuestas a interrogantes planteadas en cualquier área del conocimiento humano.

En consecuencia, la investigación hizo una revisión de diversos documentos científicos dispuestos en bases de datos indexadas de alto impacto y estudios que versaron sobre la temática de los últimos veinte años, estos resultados de la revisión bibliográfica fueron tabulados con base al análisis bibliométrico para medir la calidad e impacto de los documentos consultados sin desfavorecer la calidad del proceso investigativo de cada producto, lo que finalmente generó el sistema de categorías que es de gran importancia ya que de ella dependió la estrategia de investigación empleada.

Por consiguiente, con respecto al conocimiento científico derivado como resultado del presente estudio que concierne a la afirmación positiva de las categorías inicios: del desarrollo de software, metodologías de diseño, modelos de diseño, metodologías ágiles, desarrollo de videojuegos, suites interactivas, la integración tecnológica, SUM, entornos de desarrollo y programación, diseño algorítmico y Aplicaciones Modulares y Competencias,

creatividad e innovación en la programación en ambientes SUM a través de la aplicación del método científico, es por ello que el conocimiento generado busca explicar causalmente los fenómenos indagados por medio de las teorías ya mencionadas, y por tanto, se concibe a la razón como medio para otros fines; es decir, una razón instrumental. (Perez, 2010).

Como conclusión, para cumplir con el propósito de la investigación que corresponde a analizar la metodología SUM para el desarrollo de suite interactivos, aludiendo sus particularidades y ventajas en apoyo al desarrollo de estas herramientas, se desarrolló un estudio documental, con diseño bibliográfico descriptivo utilizando las categorías antes señaladas para proyectar un análisis desde donde poder establecer las relaciones de los tópicos de acuerdo con diferentes autores y en este sentido poder enmarcar de forma teórica la justificación acerca de la pertinencia y coherencia del despliegue de esta metodología en entornos de programación de videojuegos debido a sus capacidades y flexibilidad con respecto al equipo de trabajo.

Resultados

Suites Interactivas, la integración tecnológica

Como se ha mencionado anteriormente, el desarrollo de suite interactivas basada en videojuegos conjuga el talento de muchas disciplinas, tanto a nivel de ingeniería como a nivel artístico, en este particular para Pascarella *et al.*, (2018), una de las actividades críticas en el diseño de videojuegos es la documentación y la manipulación de sus recursos, entre ellos: arte, música, librerías y código fuente, por lo que se hace patente la necesidad de contar con un sistema integrado de desarrollo (IDE) que permita el acceso simultaneo a los múltiples entornos propios del videojuego, es decir, su arte, lógica y sonido. Un ejemplo claro de estos sistemas, son los motores de videojuego actuales que admiten al programador la visualización y manipulación de los diferentes archivos y aspectos que integraran el videojuego.

Un ejemplo de estos motores para videojuegos (Partha *et al.*, 2012), son el motor “ID Tech” empleado durante el surgimiento de los “FPS” con “DOOM” a la cabeza, el motor “SCUMM” especializado en la ejecución de aventuras gráficas como el popular juego “Maniac Mansion”, para el cual fue diseñado, o el afamado “Cry engine”, pensado para explotar al máximo la potencia gráfica de las computadoras y consolas más punteras en su tiempo, empleado en el visualmente revolucionario videojuego “Crysis”. Este paradigma inicial de desarrollar independientemente el “arte” y la “jugabilidad” del juego, para después sincronizar su ejecución en un “Engine” o “Motor de Videojuego” ha evolucionado continuamente. Hasta hoy día, los llamados “Engines” vienen acompañados de un amplio entorno de desarrollo donde es posible realizar modificaciones en texturas o parámetros de la inteligencia artificial, mostrando sus resultados en tiempo real, es así como pululan hoy en día diferentes “motores” de desarrollo pagos, gratuitos o con un modelo de negocio de “Royalties” o suscripción que integran elementos de producción y flujos de trabajo en un mismo paquete de desarrollo. Algunos casos representativos de estos motores para desarrollo de videojuegos (Christopoulou y Xinogalos, 2017) se presentan en la tabla 3:

Tabla 3

Motores de videojuegos más representativos del mercado actual

Suite	Descripción
Unity	Es un motor de videojuegos multiplataforma que está disponible para diferentes sistemas operativos como Microsoft, Linux y OS X. Permite compilar aplicaciones en diversas plataformas objetivos tales como Web, Pc, Dispositivos móviles, Smart TV, consolas y dispositivos de realidad extendida
Unreal	Es uno de los primeros motores de juego mostrado inicialmente en 1998 por la empresa Epic Games. Cuenta con características únicas como Lumens, estática mucho más realista, compatibilidad para con PS5, Xbox, Pc, Mac e IOS, entre muchas otras.
Cry engine	Este motor de gamificación, creado por la empresa Crytek que cuenta con soporte para Windows, PlayStation y Xbox 360 y con capacidad de despliegue de múltiples servicios para el desarrollo de videojuegos.
Frostbite	Es el motor gráfico desarrollado y utilizado por EA Digital Illusions CE. Sus características especiales son: Destruction 4.0, un potente motor de sonido, Tessellation Technology y el sistema de profundidad de campo.
GameMaker Studio	Compuesto por un lenguaje de programación interpretado y un SDK para el desarrollo de videojuegos, este motor fue creado por la empresa Mark Overmars y permite desarrollar fácilmente videojuegos sin necesidad de aprender un lenguaje de programación.

Fuente: elaboración propia (2019).

También es remarcable la capacidad que algunos de estos entornos tienen para facilitar en muchos casos el diseño de una única versión fuente exportable a diferentes sistemas, como por ejemplo, “HTML5” para su uso en navegadores, PC, Linux, MAC, Android, PlayStation Xbox, entre otros.

SUM, método ágil para el desarrollo de software

El desarrollo de suites interactivas basadas en videojuegos es un esfuerzo especialmente complejo a nivel metodológico, dado su carácter multidisciplinar y el alto nivel de incertidumbre y variabilidad de resultados en un proyecto de este tipo, por lo que orientar su desarrollo a través de un acercamiento sistemático y contextualizado a los objetivos propuestos, es altamente deseable (Politowski *et al.*, 2016).

El uso de metodologías ágiles en el desarrollo de videojuegos es ampliamente aceptada, sin embargo, su adaptación a las condiciones específicas de cada proyecto y en general a la industria del videojuego ha sido realizada en su gran mayoría de forma parcial, manteniendo las buenas prácticas de este tipo de metodologías solo en algunos aspectos del desarrollo, un ejemplo claro de ellos es la propuesta de Torres *et al.*, (2016), quienes segmentan el desarrollo en cuatro etapas, a saber: preproducción, producción, pruebas y post producción, sin embargo, y como es de suponer, algunos elementos

propios de este proceso requieren ajustes para adaptarse a las necesidades puntuales de cada proyecto.

Por otra parte, es posible como se mencionó anteriormente, emplear metodologías ágiles que se adaptan bastante bien a entornos cambiantes, equipos multidisciplinares y con altas cuotas de incertidumbre, por lo que metodologías con mayor trayectoria y multitud de casos de éxito como Scrum y XP, son altamente usadas en la industria del videojuego (Osborne *et al.*, 2014), sin embargo su aplicación conlleva ciertos inconvenientes como los descritos por Politowski *et al.* (2016) donde refieren acerca de objetivos poco realistas, problemas de comunicación con implicados internos y externos que no estén familiarizados con la metodología escogida y en algunos casos la dificultad de adopción de estas metodologías para grupos altamente creativos como dibujantes o guionistas como se menciona en Godoy y Barbosa (2010).

Asimismo resulta interesante, la existencia de iniciativas para generar nuevas metodologías enfocadas explícitamente al desarrollo de videojuegos como es el caso de la Metodología de desarrollo de juego ágil de agente (AAGDM) de Al-azawi, *et al.* (2014), o la metodología SUM (Acerenza *et al.*, 2009).

En este último respecto, la metodología SUM (figura 6) propuesta por Acerenza *et al.* (2009) parte de las necesidades detectadas en pequeñas empresas de desarrollo de videojuegos en el contexto latinoamericano, específicamente en Uruguay. Para su desarrollo, se toman ciertos elementos de las metodologías SCRUM y XP dada la cantidad de casos de éxito que han obtenido en el campo de los videojuegos, cuyas especificaciones y sugerencias acerca de sus buenas prácticas han sido descritas en la Especificación del metamodelo de ingeniería de procesos de software y sistemas (SPEM) de Object Management Group, (2004).

Por otra parte, esta metodología contempla cuatro roles similares a los roles descritos en SCRUM, pero caracterizándolos acorde a las necesidades del medio, estos roles son: Equipo de desarrollo, Productor interno, Cliente y Verificador beta. Asimismo, la implementación de SUM también toma en cuenta cinco ciclos iterativos para el despliegue del proyecto y un ciclo transversal a todo el desarrollo, ellos son: Concepto, Planificación, Elaboración, Beta y Cierre. Asimismo, con los roles asignados y las buenas prácticas sugeridas para el proceso de desarrollo, aporta múltiples casos de éxito a través de la literatura consultada tal es el caso de los “Juegos Serios”, conocidos en inglés como “Serious Games”, los cuales buscan aprovechar las vertientes lúdicas e inmersivas asociadas a los videojuegos como por ejemplo, las realizadas por Molero *et al.* (2019) y Pineda y Estefanía, (2018) y Arambarri *et al.* (2012).

Figura 6
Metodología SUM

Fuente: elaboración propia (2019).

En adición a lo anterior, buenas prácticas orientadas al desarrollo de juegos serios concierne al despliegue de suite interactivas basadas en videojuegos en apoyo al proceso de rehabilitación, cuyo fin ulterior es el de impactar positivamente en ambientes más allá del entretenimiento haciendo más motivador en inclusivo este tipo de ejercicios en múltiples dolencias tanto osteomusculares, como psíquicas e incluso en el tratamiento de anomalías neuronales (Molero *et al.*, 2019).

Por contraste, a pesar de encontrarse en la categoría de “juegos serios”, su aplicación junto a dispositivos periféricos como Kinect y Leap motion, entre otros (Ferrarini *et al.*, 2018), han conseguido en la metodología SUM un muy importante medio para el despliegue de bastas soluciones.

Diseño Algorítmico y Aplicaciones Modulares

El diseño de algoritmos que da solución a problemas diversos se ejecuta mediante un proceso de manipulación de datos, beneficio producto de la aplicación de conocimientos, experiencia del programador y creatividad, sin embargo, las ciencias de la computación han aportado múltiples técnicas que permiten fortalecer aspectos concretos de la programación (Oviedo, 2004). Un ejemplo claro de estas técnicas corresponde a los algoritmos paralelos, quienes reducen la solución de un problema en la ejecución de pequeñas tareas que se pueden realizar de forma independiente; los algoritmos obtenidos mediante esta técnica son de alto desempeño cuando se emplean múltiples núcleos de procesamiento.

En otro particular, la técnica conocida como “divide y vencerás” (Goodrich, 2014) consiste en dividir un gran proceso en varios módulos que a su vez pueden ser reducidos a módulos más pequeños, hasta poder formar un algoritmo en pseudocódigo fácil de implementar en código fuente. Estos módulos tienen la capacidad de ser reutilizables reduciendo la necesidad de generar nuevos algoritmos.

Es entonces, cuando es común que los motores para desarrollo de videojuegos incluyan entre sus funciones módulos y “assets” que faciliten su desarrollo en géneros específicos, como juegos de carreras, “FPS”, e incluso juegos en 2D o 3D. Estos módulos enlazables entre sí pueden ser modificados y personalizados para ser reutilizados en futuros proyectos e incluso licenciarse o compartirse con terceros, como por ejemplo, los motores de desarrollo Unity, Unreal y Cryengine que cuentan con tiendas en línea y comunidades de desarrolladores que ofrecen módulos para expandir las capacidades tanto de la jugabilidad, como del mismo editor.

Competencias, creatividad e innovación en la programación en ambientes SUM

La creatividad es un proceso cognitivo complejo en el que interviene una amplia cantidad de factores, sobre los cuales no existe consenso a través de la comunidad científica, sin embargo, hay ciertas actitudes comunes en una gran cantidad de creativos (Chavarria, 2016), actitudes entre las cuales se destacan: carácter independiente de pensamiento y acción, gusto por la complejidad y los fenómenos inconclusos, gran capacidad de relacionar conceptos y temáticas disímiles, gran confianza y capacidad de decisión, entre otras tantas.

Estas actitudes creativas, han sido en gran parte el motor para el notable crecimiento de la industria del videojuego y su trascender a otras áreas a través de la ludificación, a pesar que en la mayoría de casos, los estudios creativos que buscan generar su propio estilo en el medio muy frecuentemente dependen de empresas editoras y de distribución que buscan cierto nivel de seguridad en el retorno de su inversión, donde, una idea creativa por sí sola, no genera confianza en el producto final. El siguiente escalón es el enfoque de la innovación, en el cual se formaliza el aprovechamiento del proceso creativo centrando su esfuerzo en el producto (Ted, 2003), es decir, emplear el pensamiento creativo no en la búsqueda de conceptos de juego revolucionarios, sino en aplicar los recursos disponibles en la consecución de los objetivos propuestos.

Los recursos de un proyecto si bien contemplan equipos de cómputo, propiedad de licencias, software y otros activos, también incluyen activos intangibles que suelen ser la diferencia entre un proyecto exitoso y uno fallido. En ese sentido, el capital intelectual (Ospina *et al.*, 2019) que compone el equipo de desarrollo, particularmente en el marco de la creación de videojuegos, debe poseer competencias específicas para cada rama del proceso de gamificación, dependiendo del tipo de proyecto a desarrollar. En ese particular, (Ted, 2003) afirma que importante destacar las competencias en los siguientes campos: Ingeniería de software, programación, artes plásticas, ingeniería de audio, animadores, narrativa, diseño de niveles, auditoria y testeo de calidad, mientras que otros estudios más informales como (Quora, 2017) y (Oxford College, 2019), decantan competencias blandas como: originalidad, análisis de sistemas, pensamiento crítico, manejo de tiempo, aprendizaje activo, visualización, y atención selectiva.

Extrapolando estas necesidades desde la creación de los videojuegos al despliegue de la metodología SUM, se pueden definir una serie de competencias blandas para cada rol de esta metodología, las cuales se muestran a continuación:

Equipo de desarrollo: sub-roles de:

- Programador: análisis de problemas, análisis numérico, delegación, tolerancia al estrés, planificación y organización, tenacidad y trabajo en equipo.
- Artista gráfico: creatividad, flexibilidad, comunicación verbal y no verbal, adaptabilidad, trabajo en equipo, planificación y organización.
- Artista sonoro: creatividad, flexibilidad, comunicación verbal y no verbal, adaptabilidad, trabajo en equipo, planificación y organización.
- Diseñador de juego: análisis de problemas, creatividad, capacidad crítica, sociabilidad, trabajo en equipo, planificación y organización, análisis de sistemas.
- Productor interno: liderazgo, delegación, sociabilidad, análisis de sistemas, trabajo en equipo, integridad, iniciativa, análisis de problemas, atención al cliente y asunción de riesgos.
- Cliente: Si bien el cliente es un actor externo al equipo de desarrollo, si es deseable que cuente con las siguientes competencias: capacidad crítica, comunicación verbal y no verbal, asunción de riesgos, integridad y sensibilidad interpersonal y organizacional.
- Verificador beta: integridad, meticulosidad, comunicación verbal y no verbal, trabajo en equipo, compromiso y capacidad crítica.
- Como se mencionó anteriormente, si bien estas competencias blandas son deseables en los roles para el desarrollo de la mayoría de los proyectos, esto depende en gran medida de las características propias de cada proyecto, por lo cual esto es solo una observación resultante de los múltiples estudios que orientaron el desarrollo de este capítulo.

Conclusiones

El desarrollo de suites interactivas basada en videojuegos ofrece múltiples escenarios de aplicación como es la educación, la rehabilitación, el entrenamiento y el mercadeo, entre muchos otros, de forma directa a través de estrategias específicas o de forma indirecta, aprovechando los sistemas gamificados existentes en un contexto particular. En consecuencia, su éxito o fracaso depende de múltiples factores que incluyen las metodologías que amalgaman todos los talentos del proyecto dado que, si bien la creatividad conlleva a ideas novedosas e interesantes, es su desarrollo integral, sus mecánicas, arte y programación correctamente orquestadas, lo que hace exitoso un videojuego.

En adición se destaca que un equipo de diseño para videojuegos requiere una metodología que reduzca los fallos y evite que se den vueltas en círculo sobre algún problema en particular, las metodologías altamente secuenciales o en cascada, generan tiempos muertos para algunos equipos de trabajo, al tener que esperar la culminación de la tarea precedente, que en ocasiones

por falta de coordinación requiere ser realizada de nuevo al no satisfacer las necesidades del proyecto o ser incompatible con el trabajo del resto de equipo de desarrollo.

Las metodologías ágiles en contraste, ofrecen una aproximación interesante a la gestión de desarrollo en el ámbito de la ludificación, sin embargo, su adaptación a las particularidades de cada equipo y proyecto requieren de personal altamente capacitado y con amplia experiencia en el apartado metodológico y de dirección, por ende, la adopción de una metodología específicamente pensada para el ámbito lúdico, facilita en gran medida la adopción de buenas prácticas y una coordinación adecuada para el proyecto.

Estas inferencias llevan a concluir, que la metodología SUM discurrida desde una perspectiva inicial de éxito como SCRUM sumado al desarrollo iterativo basado en sistemas gamificados, muestra roles coherentes con las necesidades observadas en la literatura consultada y el propósito de la investigación, que reviste el analizar esta metodología para el desarrollo de suite interactivas, aludiendo sus particularidades y ventajas en apoyo al desarrollo de esta tecnología y por si fuera poco, el flujo de trabajo propuesto por esta sistematización muestra ser compatible con las competencias y capacidades del equipo de desarrollo dada la flexibilidad en la asignación de tareas descritas en su definición y práctica.

Referencias bibliográficas

- Acerenza, N., Coppes, A., Mesa, G., Viera, A., Fernández, E., Laurenzo, T., y Vallespir, D. (2009). *Una metodología para desarrollo de videojuegos*. 38° JAIIO - Simposio Argentino de Ingeniería de Software (ASSE 2009), (págs. 171-176). https://www.fing.edu.uy/sites/default/files/biblio/22811/asse_2009_16.pdf
- Al-azawi, R., Ayesh, A., y Al. Obaidy, M. (2014). *Towards Agent-based Agile approach for Game Development Methodology*. 2014 World Congress on Computer Applications and Information Systems (WCCAIS). doi:DOI: 10.1109/WCCAIS.2014.6916626
- Arambarri, J., Armentia, L., y Baeza, U. (2012). Serious games para la puesta en valor de la cultura. Un caso práctico: SUM. *Virtual Archaeology Review*, 3(7), 65-67. doi:<https://doi.org/10.4995/var.2012.4388>
- Benito, J. (2006). *El mercado del videojuego. Unas cifras*. Icono14, 4(1). <https://dialnet.unirioja.es/servlet/articulo?codigo=2043889>
- Booch, G. (1993). *Object-Oriented Analysis and Design with Applications (OBT)*. Memphis, TN, Estados Unidos de America: Benjamin-Cummings Publishing Company.
- Booch, G. (2018). *The History of Software Engineering*. IEEE Software, 35(5), 108-114. doi:DOI: 10.1109/MS.2018.3571234
- Chavarria, M. (2016). La eficacia de la creatividad Creactivate. Barcelona, España: ESIC.
- Christopoulou, E., y Xinogalos, S. (2017). Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices. *International Journal of Serious Games*, 4(4), 21-36. doi:<https://doi.org/10.17083/ijsg.v4i4.194>

- Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Boston, United States of America: Pearson Education, Inc.
- Ferrarini, C., González, M., Olguín, L., y Ormeño, E. (2018). *Realidad virtual en terapias para estimular la cognición de niños con encefalopatía crónica no evolutiva. XX Workshop de Investigadores en Ciencias de la Computación*. (págs. 1111-1114). Argentina. <http://sedici.unlp.edu.ar/handle/10915/68400>
- Fowler, M., Beck, K., Roberts, D., Brant, J., y Opdyke, W. (1999). *Refactoring: Improving the Design of Existing Code*. Massachusetts: C4Media Inc.
- Godoy, A., y Barbosa, E. (2010). *Game-Scrum: An Approach to Agile Game Development*. SBC - Proceedings of SBGames, 292-295. http://sbgames.org/papers/sbgames10/computing/short/Computing_short19.pdf
- Goodrich, M. (2014). *Algorithm design and applications*. United States of America: Wiley.
- Hall, J., Stickler, U., Herodotou, C., y Iacovides, I. (2020). Expressivity of creativity and creative design considerations in digital games. *Computers in Human Behavior*, 105, 1-40. doi:<https://doi.org/10.1016/j.chb.2019.106206>
- Hernández, R., Fernández, C., y Baptista, P. (2014). *Metodología de la Investigación*. 6ta Edición. México: McGrawHill.
- ISO/IEC 19501. (2005). *ISO/IEC 19501:2005 Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*. ISO/IEC, 432. <https://www.iso.org/standard/32620.html>
- ISO/IEC/IEEE 42010. (2011). *Systems and software engineering — Architecture description*. ISO/IEC/IEEE, 37. <https://www.iso.org/standard/50508.html>
- Jacobson, I., Christerson, M., Johnsson, P., y Overgaard, G. (1993). *Object-Oriented Software Engineering: A Use Case Driven Approach*. New York, United States of America: (ACM Press) Addison-Wesley.
- Kendall, K., y Kendall, J. (2005). *Análisis y diseño de Sistemas*. México: Pearson Educación.
- Kniberg, H., y Skarin, M. (2010). *Kanban and Scrum - making the most of both*. United States of America.
- Kniberg, H. (2007). *Scrum and XP from the Trenches*. United States of America : C4Media Inc.
- Kruchten, P. (1995). Architectural Blueprints—The “4+1” View Model of Software Architecture. *IEEE Software*, 12(6), 42-50. <https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf>
- Kuo, A., Hiler, J., y Lutz, R. (2017). From Super Mario to Skyrim: A framework for the evolution of video game consumption. *Journal of Consumer Behaviour*, 101-120. doi:<https://doi.org/10.1002/cb.1620>
- López, C. (2016). El videojuego como herramienta educativa. Posibilidades y problemáticas acerca de los serious games. *Apertura*, 8(1), 1-15. <http://www.udgvirtual.udg.mx/apertura/index.php/apertura/article/view/825>

- Mall, R. (2018). *Fundamentals of software engineering*. Delhi: PHI Learning.
- McConnell, S. (2004). *Code Complete: A Practical Handbook of Software Costruction*. United States of America: Microsoft Press.
- Molero, L., Batista, J., y Puentes, M. (2019). Transformación digital desde la cultura tecnológica en las instituciones de educacion superior del estado zulia. *Consensus*, Santiago de Chile, Chile.
- Molero, L., Díaz, P., Montenegro, J., Castiblanco, L., y Ruiz, J. (2019). Usabilidad de interfaz 3D móvil desarrollada para HMD dirigida a usuarios con movilidad reducida. En L. Molero Suárez, P. Díaz Montaña, J. Montenegro Romero, L. Castiblanco Rosero, & J. Ruiz Carrillo. *Gamificación y discapacidad. Una alternativa socialmente responsable*, Volumen I. Santa Ana de Coro, Venezuela: Fondo Editorial Universitario Servando Garcés de la Universidad Politécnica Territorial de Falcón Alonso Gamero.
- Molero, L., Ospina, P., Ayala, E., y Lozano Rojas, H. (2019). Discapacidad motora y la rehabilitación desde los ambientes virtuales basados en kinect. En L. Molero Suárez, P. Ospina Rodriguez, E. Ayala Escobar, & H. Lozano Rojas. *Tendencias en la Investigación Universitaria: Una visión desde Latinoamérica*. (págs. 174-186). Bogotá Colombia: Fondo Editorial Universitario Servando Garcés de la Universidad Politécnica Territorial de Falcón Alonso Gamero.
- Molero, L., Ospina, P., Quevedo, M., Montenegro, J., y Castiblanco, L. (2019). Gaming, una estrategia para la rehabilitación de personas con discapacidad. En L. Molero Suárez, P. Ospina Rodriguez, M. Quevedo, J. Montenegro, & L. Castiblanco. *Tendencias en la Investigación Universitaria: Una visión desde Latinoamérica*, Volumen VI (págs. 134-144). Santa Ana de Coro, Venezuela: Fondo Editorial Universitario Servando Garcés de la Universidad Politécnica Territorial de Falcón Alonso Gamero.
- Molero, L., Sáenz, J., Montenegro, J., Castiblanco, L., y Ruiz, J. (2019). Interfaz 3D móvil con HMD para personas con discapacidad motora apoyada en metodología SUM. En L. Molero Suárez, J. Sáenz Leguizamón, J. Montenegro Romero, L. Castiblanco Rosero, & J. Ruiz Carrillo. *Gamificación y discapacidad. Una alternativa socialmente responsable*, Volumen I. Santa Ana de Coro, Venezuela: Fondo Editorial Universitario Servando Garcés de la Universidad Politécnica Territorial de Falcón Alonso Gamero.
- Nava, H., (2002). *La investigación jurídica. ¿Cómo se elabora el proyecto?* Maracaibo. Editorial EDILUZ.
- Object Management Group. (2004). *Software & Systems Process Engineering Meta-Model Specification*. <https://www.omg.org/spec/SPEM/2.0/PDF>
- Osborne, A., Coleman, G., y O'Connor, R. (2014). Software Development Processes for Games: A Systematic Literature Review. In: Barafort B., O'Connor R.V., Poth A., Messnarz R. Systems, Software and Services Process Improvement. *EuroSPI* 2014., 425, 182-193. doi:https://doi.org/10.1007/978-3-662-43896-1_16

- Ospina, P., Molero, L., Lozano, H., Becerra, Y., y Ruiz, J. (2019). Capital intelectual y responsabilidad social en entornos de programación de realidad virtual. En P. E. Ospina Rodríguez, L. Molero Suárez, H. Lozano, Y. Becerra, & J. Ruiz. *Tendencias en la Investigación Universitaria: Una visión desde Latinoamérica*, Volumen VII. Santa Ana de Coro, Venezuela: Fondo Editorial Universitario Servando Garcés de la Universidad Politécnica Territorial de Falcón Alonso Gamero.
- Oviedo, F. (2004). *Diseño estructurado de algoritmos*. Colombia.
- Oxford College. (2019). *Oxford College*. <https://www.oxfordedu.ca/blog/4-essential-skills-youll-need-for-a-video-game-designer-career>
- Partha Sarathi , P., Surajit , G., y Abhishek , B. (2012). History and comparative study of modern game engines. *International Journal of Advanced Computer and Mathematical Sciences*, 3(2), 245-249. https://www.researchgate.net/publication/259496289_History_and_comparative_study_of_modern_game_engines
- Pascarella, L., Palomba, F., Di Penta, M., y Bacchelli, A. (2018). *How Is Video Game Development Different from Software Development in Open Source?*, 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR). IEEE, 392-402.
- Perez. (2010). *Enfoques metodológicos*. Editorial La Salle.
- Pineda, C., y Estefanía, K. (2018). *Difusión de conocimientos histórico-culturales de la batalla de Ibarra mediante un videojuego dirigido a los niños de 5to año de educación básica de la unidad educativa “Daniel Reyes” de la ciudad de Ibarra* (Tesis de Pregrado). Ecuador. <http://repositorio.utn.edu.ec/handle/123456789/8649>
- Politowski, C., Vargas, D., Fontoura, L., y Foletto, A. (2016). Software Engineering Processes in Game Development: a Survey about Brazilian Developers' Experiences. SBGames 2016.
- Quora. (2017). *Forbes*. <https://www.forbes.com/sites/quora/2017/09/01/what-skills-do-game-designers-need/#2ec63c818536>
- Rumbaugh, J. (1996). *Modelado y diseño orientados a objetos: Metodología OMT*. España: Prentice Hall.
- Rumbaugh, J., Jacobson, I., y Booch, G. (2000). *El lenguaje Unificado de Modelado*. Manual de referencia. Madrid, España: Addison Wesley.
- Schell, j. (2019). *The Art of Game Design A Book of Lenses, Third Edition*. New York, United States of America: A K Peters/CRC Press. doi:<https://doi.org/10.1201/b22101>
- Ted, F. (2003). *When Does an Idea Become an Innovation?* The Role of Individual and Group Creativity in Videogame Design. Singapore: Research Collection Lee Kong Chian School of Business.

- Torres-Ferreiros, C., Festini-Wendorff, M., y Shiguihara-Juárez, P. (2016). *Developing a videogame using unreal engine based on a four stages methodology*. 2016 IEEE ANDESCON. doi:<https://doi.org/10.1109/ANDESCON.2016.7836249>
- Wilkinson, P. (2016). A Brief History of Serious Games. In: Dörner R., Göbel S., Kickmeier-Rust M., Masuch M., Zweig K. (eds) *Entertainment Computing and Serious Games*. Lecture Notes in Computer Science (Vol. 9970). Springer, Cham. doi:https://doi.org/10.1007/978-3-319-46152-6_2
- Zachman, J. (1987). *A framework for information systems architecture*. IBM Systems Journal, 276 - 292. doi:DOI: 10.1147/sj.263.0276