Universidad de las Ciencias Informáticas

Facultad 3

Módulo de Producción del Simulador con Fines Docentes (SINEG)

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Osiel Sánchez Martínez

Tutores:

MsC. Elizabeth Rodríguez Stiven
Ing. Dailien Moré Soto
Ing. Leodanny Wuilber Polanco Garay

Junio, 2017.

"Año 59 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____

Osiel Sánchez Martínez

Autor

_____ _____

MsC. Elizabeth Rodríguez Stiven
Tutor

Ing. Dailien Moré Soto
Tutor

_____

Ing. Leodanny Wuilber Polanco Garay

Tutor

Datos de Contacto

MsC. Elizabeth Rodríguez Stiven
Ciudadanía: cubana.
Institución: Universidad de las Ciencias Informáticas, La Habana, Cuba.
E-mail: beth@uci.cu

Ing. Dailien Moré Soto
Ciudadanía: cubana.
Institución: Universidad de las Ciencias Informáticas, La Habana, Cuba.
E-mail: dmore@uci.cu

Ing. Leodanny Wuilber Polanco Garay
Ciudadanía: cubana.
Institución: Universidad de las Ciencias Informáticas, La Habana, Cuba.
E-mail: lwpolanco@uci.cu

AGRADECIMIENTOS

A la Revolución y al Comandante en Jefe Fidel Castro Ruz por darme la posibilidad de formarme como digno profesional,

A la UCI por permitirme crecer en todos los sentidos posibles,

A mis padres y familiares por ser el apoyo necesario,

A mis amigos, los verdaderos, por existir y acompañarme en esta travesía,

A Dailien, por ser, por estar, por sentir.

DEDICATORIA

A mi madre, la que he amado desde siempre,

A mi abuelo, que siempre está a mi lado,

A mi abuela, la persona que más amo en vida,

A mi papá, un ejemplo a seguir siempre.

RESUMEN

Un simulador es un programa de computación que permite poner en práctica los conocimientos adquiridos en la escuela. Esta acción virtual de negociar, propone la posibilidad de enfrentarse a situaciones reales y tomar decisiones que pueden llegar a afectar o beneficiar a la empresa. En el presente trabajo se describe el proceso de desarrollo del módulo de producción de un simulador de negocios para los estudiantes de la carrera de Ingeniería Industrial en la CUJAE, que les sirva para estimar la producción terminada aprovechando los factores productivos. Con este objetivo se realizó un estudio del estado del arte de la simulación de negocio para identificar las buenas prácticas empleadas en los simuladores que estiman la producción empresarial. Se representó la arquitectura definida para la aplicación, así como, los requisitos funcionales y no funcionales que rigieron el proceso de desarrollo de la misma, concluyendo con un conjunto de pruebas que validó su funcionamiento. Como resultado se obtuvo un módulo de producción del simulador de negocios con fines docentes, que permite estimar la producción terminada de las empresas teniendo en cuenta los factores productivos.

PALABRAS CLAVES: capacidad, estimación, jornada laboral, mantenimiento, negocio, producción, simulador .

ABSTRACT

A simulator is a computer program that allows putting into practice the knowledge acquired in school. This virtual action of negotiation, proposes the possibility of facing real situations and making decisions that can affect or benefit the company. This paper describes the process of developing the production module of a business simulator for the students of the Industrial Engineering career in the CUJAE, which will help them to estimate the finished production taking advantage of the productive factors. With this objective a study of the state of the art of the business simulation was carried out to identify the good practices used in the simulators that estimate the business production. The architecture defined for the application was represented, as well as the functional and non-functional requirements that guided the development process of it, concluding with a set of tests that validated its performance. As a result, a production module of the business simulator for teaching purposes was obtained, which allows to estimate the finished production of the companies taking into account the productive factors.

KEYWORDS: capacity, estimate, working day, maintenance, business, production, simulator.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Los simuladores de negocios o juegos de negocios son herramientas de apoyo en el proceso de aprendizaje, dado que permiten establecer un ambiente virtual de negocios a fin que los estudiantes tengan la oportunidad de participar, a través de un conjunto de decisiones, en el proceso de dirección de una empresa o de un área específica de la misma**[1]** .

En su mayoría son programas de computación que se construyen usando un lenguaje de programación. Dichos programas son elaborados considerando tanto la relación que existe entre los factores internos de operación de una empresa, así como de algunas variables del entorno que la afectan en su operación **[2]**.

Se puede decir que los simuladores de negocios son modelos que se construyen a partir de especificar un número de variables relevantes internas y también externas, las cuales deben permitir simular la operación de una empresa en un contexto cambiante y de competencia con otras compañías similares **[3]**.

El uso de simuladores de negocios, cada vez más extendido mundialmente tanto en la enseñanza académica como en la capacitación empresarial, tiene como objetivo cubrir la brecha entre teoría y práctica que se daba en la educación tradicional**[4]**.

Con estas herramientas, los alumnos pueden aprender sin peligro de destruir recursos reales y con el beneficio extra de poder condensar, en el tiempo que dura un semestre universitario, procesos que en el mundo real demorarían años en producirse.

El uso de simuladores de negocios constituye una actividad esencialmente grupal, ya que cada equipo de alumnos se encuentra al frente de una empresa que compite con las demás empresas, conformando así un mercado. Mundialmente, universidades de alto prestigio como Stanford y Carnegie Mellon, hacen uso de las TIC para la generación de contenidos educativos, entre ellas destacan los simuladores, como un recurso que pone al alumno en contacto con la realidad de los contenidos **[5]**. En Cuba, no se muestran evidencias sobre el uso de simuladores con fines docentes dedicados a la gestión empresarial, sin embargo, ya existen estudios sobre el tema en investigaciones del ámbito académico.

En el Instituto Superior Politécnico José Antonio Echevarría (CUJAE) en la carrera de Ingeniería Industrial se imparte la asignatura Simulación de Negocio, entre sus temáticas se encuentran los negocios de ensamblaje de productos. Esta asignatura tiene como objetivo fundamental desarrollar habilidades de toma de decisiones en el ámbito empresarial. En la actualidad esto se ha visto limitado, puesto que no logran reflejar el ambiente deseado al realizar un engorroso proceso de obtención y manipulación de datos en el espacio de tiempo, que, a su vez, es la fuente para la interpretación de las soluciones arrojadas y las posibles decisiones a tomar. Además, solo se realizan ejercicios académicos de poca complejidad que distan de la realidad empresarial encargada de simular lo procesos del subsistema productivo, dándose el caso de usar variables fijas en los ejercicios y contando siempre con pocas líneas de producción u otros componentes de la empresa. No obstante, la principal deficiencia identificada en la asignatura es que para la realización de ejercicios académicos no aprovechan satisfactoriamente los datos asociados a factores productivos tales como: materias primas y materiales, recursos humanos y tecnologías.

La situación problémica antes descrita genera el siguiente **problema de investigación** ¿Cómo estimar la producción terminada de manera que se aprovechen los datos asociados a los factores productivos?; tomando en cuenta el problema antes propuesto se define como **objeto de estudio:** el desarrollo de software para la gestión empresarial, identificándose como **campo de acción:** Los sistemas informáticos para la estimación de la producción terminada. A su vez se determina como **objetivo general:** Desarrollar el Módulo de Producción del Simulador de Negocio SINEG para la estimación de la producción terminada que aproveche los factores productivos.

Se desglosan del objetivo general los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación, mediante el estudio de los referentes teóricos de los simuladores de negocio, metodología de desarrollo, técnicas y herramientas de software.
- Modelar matemáticamente **la producción terminada** de la empresa utilizando los indicadores Coeficiente de Disponibilidad Técnica (CDT) y Aprovechamiento de la Jornada Laboral (AJL).

- Implementar el módulo de producción a partir del modelo matemático definido para el cálculo de los factores productivos.
- Valorar la solución propuesta mediante la viabilidad del software y el nivel de aceptación del cliente.

Para dar cumplimiento a los objetivos específicos planteados se definen las siguientes **tareas de la investigación:**

- Desarrollar el marco teórico referencial de la investigación en función de los términos relacionados con los factores productivos de la empresa.
- Caracterizar las herramientas a utilizar en la investigación.
- Identificar y validar los requisitos funcionales y no funcionales a tener en cuenta en el desarrollo de la solución.
- Definir la arquitectura a utilizar en el desarrollo del trabajo.
- Definir el modelo matemático a tener en cuenta para el cálculo de los factores productivos.
- Implementar el módulo a partir del modelo matemático definido para el cálculo de los factores productivos.
- Realizar las pruebas de validación a la herramienta informática, aplicando métodos y métricas.
- Realizar el informe final de la investigación.

Determinándose como **idea a defender:**

El desarrollo del Módulo de Producción del Simulador de Negocio SINEG permitirá la estimación de la producción terminada aprovechando los factores productivos.

Para el análisis del negocio y la obtención de información se utilizaron los siguientes métodos teóricos:

Histórico-lógico se empleó para profundizar en las tendencias y tratamientos históricos de la simulación con fines educativos y determinar su influencia en la estimación de la producción terminada.

Analítico – sintético se empleó para el procesamiento de la información y arribar a conclusiones prácticas y teóricas de la investigación. A partir del análisis de los referentes teóricos y la bibliografía en cuestión, se realizó una síntesis de los elementos significativos de la investigación, entre ellos el análisis de los simuladores existentes y sus buenas prácticas.

Inductivo-deductivo se empleó en el desarrollo de la investigación, mediante la inducción se realizó un razonamiento que partió de lo particular a lo general, obteniendo como resultado final las semejanzas existentes en cada una de las bibliografías estudiadas.

Modelación permitió representar los modelos y diagramas de cada una de las fases de desarrollo para apoyar la investigación.

Métodos empíricos:

Observación se utilizó para realizar una evaluación de la situación problemática en cuestión, observando el desarrollo del proceso antes de realizar la investigación.

Para mostrar una correcta organización se **estructura la investigación** de manera tal que el contenido de este trabajo consta de tres capítulos, definidos de la siguiente forma:

Capítulo 1: Fundamentación Teórica

Se presentan un conjunto de conceptos relacionados con la problemática descrita con anterioridad. Además, se realiza un estudio del estado del arte de las herramientas informáticas destinadas a la gestión empresarial y entre ella las que determinan la producción terminada de las empresas. Así como también un análisis de las metodologías, herramientas y lenguajes de programación que aporten a la investigación, determinándose a su vez las que se deben emplear en su desarrollo.

Capítulo 2: Análisis, diseño e implementación de la herramienta informática

Se describen las principales características de la herramienta propuesta, así como el diseño de la misma, guiando todo el proceso por las fases definidas en la metodología XP (Programming Extreme o Programación Extrema). Se definen los componentes necesarios para conformar un ambiente empresarial y determinar la producción terminada de las empresas, teniendo en cuenta los factores productivos. Se define el diseño de la base de datos empleada y se elaboran los elementos necesarios para garantizar el éxito en el proceso de desarrollo, determinados por la metodología, tales como: las historias de usuario, arquitectura y patrones de diseño.

Capítulo 3: Pruebas de la herramienta informática.

Se da continuidad a las fases de la metodología XP descritas en los capítulos anteriores, con la realización de pruebas unitarias y pruebas de aceptación con el cliente a la herramienta propuesta, describiéndose los principales artefactos generados, tales como las actas de liberación interna de productos de software y de aceptación del cliente. Además, se describen los resultados de la aplicación de las métricas para validar el diseño y las técnicas para la validación de los requisitos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se describen los conceptos asociados al dominio del problema, se hace un análisis del estado actual en que se encuentran las herramientas para determinar la producción terminada en las empresas. Se analizan las metodologías del proceso de desarrollo de software, las herramientas de ingeniería del software asistida por computadora (CASE), los lenguajes de programación, los entornos de desarrollo integrado, los sistemas gestores de bases de datos, con el propósito de definir las tecnologías que se emplean en el desarrollo de la herramienta. Además, se presenta un estudio de las tendencias actuales y las técnicas existentes para el uso de simuladores de negocios con fines educativos.

1.2 Marco teórico de la investigación

Para comenzar el desarrollo de la investigación es necesario conocer el término simulación, a continuación, se presentan algunas definiciones dados por autores que sirvieron de referencia para el análisis del tema.

"Simulación es una técnica numérica para conducir experimentos en una computadora digital. Estos experimentos comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real a través de largos periodos de tiempo" **[6]**.

"La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con el mismo con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias, dentro de los límites impuestos por un cierto criterio o un conjunto de ellos, para el funcionamiento del sistema" **[7]**.

Después de un análisis bibliográfico sobre los conceptos asociados a los simuladores se decide asumir como concepto a emplear durante el desarrollo de la investigación:

Un simulador puede ser un dispositivo o aparato que simula un fenómeno, el funcionamiento real de un aparato, dispositivo o las condiciones de entorno a la que está sometidos una máquina, posibilitan un aprendizaje significativo por descubrimiento y la investigación, según el simulador, puede realizarse en tiempo real o en tiempo acelerado **[8]**.

A continuación, se presentan otros conceptos que son de imprescindible entendimiento para el desarrollo de la investigación.

La **capacidad instalada** es el potencial de producción o volumen máximo de producción que una empresa en particular, unidad, departamento o sección; puede lograr durante un período de tiempo determinado, teniendo en cuenta todos los recursos que tienen disponibles, sea los equipos de producción, instalaciones, recursos humanos, tecnología, experiencia/conocimientos, etc. **[8]**.

Se denomina **producción** a cualquier tipo de actividad destinada a la fabricación, elaboración u obtención de bienes y servicios. La producción de una empresa puede medirse en un determinado volumen. La diferencia entre el volumen de lo producido en términos de dinero, en relación a los bienes consumidos, da cuenta del valor que se ha añadido a esos recursos. Así, según la diferencia que se haga de la utilización de los factores de producción con respecto a los valores de producción final se tendrá referencia a la rentabilidad o ganancia de la organización comercial **[2]**.

La **productividad laboral** es un indicador de eficiencia que se obtiene de la relación entre el producto obtenido y la cantidad insumos laborales invertidos en su producción. La productividad es la relación entre el resultado de una actividad productiva y los medios que han sido necesarios para obtener dicha producción. En el campo empresarial se define la productividad empresarial como el resultado de las acciones que se deben llevar a término para conseguir los objetivos de la empresa y un buen clima laboral, teniendo en cuenta la relación entre los recursos que se invierten para alcanzar los objetivos y los resultados de los mismos **[9]**.

El **mantenimiento industrial** es uno de los ejes fundamentales dentro de la industria, está cuantificado en la cantidad y calidad de la producción. El mismo ha estado sujeto a

diferentes cambios al paso del tiempo, en la actualidad, se ve como una inversión que ayuda a mejorar y mantener la calidad en la producción. Se define como un conjunto de normas y técnicas establecidas para la conservación de la maquinaria e instalaciones de una planta industrial, para que proporcione mejor rendimiento en el mayor tiempo posible **[10]**.

Se define **cuello de botella** como un proceso (dentro de los muchos procesos interrelacionados de la empresa), que tiene una capacidad inferior al resto y ésta no alcanza para dar curso a lo que los clientes requieren **[11]**.

Se consideran **factores productivos** a todo recurso requerido para producir bienes y servicios; dígase materia prima, recursos humanos y tecnología **[11]**.

Se denomina **recursos humanos** a las personas con las que una organización (con o sin fines de lucro, y de cualquier tipo de asociación) cuenta para desarrollar y ejecutar de manera correcta las acciones, actividades, labores y tareas que deben realizarse y que han sido solicitadas a dichas personas **[8]**.

Los **recursos materiales** son los bienes tangibles que la organización puede utilizar para el logro de sus objetivos. En los recursos materiales podemos encontrar los siguientes elementos: maquinarias, inmuebles, insumos, productos terminados, instrumentos y herramientas **[7]**.

Una **línea de producción** es el conjunto armonizado de diversos subsistemas como son: neumáticos, hidráulicos, mecánicos, electrónicos, software, etc. Todos estos con una finalidad en común: transformar o integrar materia prima en otros productos **[11]**.

1.3 Tendencias actuales de los simuladores de negocio

Diferentes autores han coincidido en que los simuladores de negocios:

- Permiten la aplicación del conocimiento a la solución de problemas.
- Mejoran la transferencia y retención de conocimientos.
- Aumentan la comprensión de conceptos abstractos y la motivación de los alumnos.
- Nivelan hacia arriba; es decir, son más efectivos con aquellos alumnos que tuvieron un bajo rendimiento previo a la simulación.

Los simuladores de negocios se pueden clasificar como:

Generales: cuando están orientados a mostrar el uso de las estrategias a nivel de negocios y las principales decisiones que debe tomar la dirección general de una empresa. Entre los principales tenemos al Business Policy Game, *Business Strategic Game*, CEO, Threshold y el Multinational Management Game.

Específicos: cuando están enfocados a simular las actividades de un área específica de una empresa como marketing, finanzas y producción. Entre los principales simuladores de este tipo tenemos al: Markstrat, Brandmaps, Marketplace, Shoes: A marketing game y Marketing Simulation, los cuales están orientados a simular las actividades de marketing; Fingame: para el área de finanzas; Forad: enfocado al área de finanzas internacionales; Intopia: para los negocios internacionales; The Management / Accounting Simulation: para el área de contabilidad **[2]**.

Para el desarrollo de esta investigación se analizaron los simuladores tanto específicos como generales que contemplaran el área de Producción, entre los que destacan:

Simulador Global2020 reproduce el escenario competitivo de cinco compañías productoras de artículos textiles de vestir, que deben competir entre sí. Todas ellas empiezan la simulación en la misma posición competitiva. Las compañías producen y comercializan tres líneas de producto: camisas, corbatas y complementos de vestir.

La compañía está fabricando y comercializando sus productos en un país, pero en el desarrollo de la simulación puede decidir implantar una fábrica en otro país y entrar en nuevos mercados. Cada mercado está definido por tres segmentos de demanda. Utiliza el factor de mantenimiento como unos de los indicadores de productividad **[12]**.

Simulador TechCompany reproduce el escenario competitivo de cinco compañías de hardware y software que deben competir entre sí. Estas comienzan la simulación con las mismas condiciones competitivas. Las compañías producen y comercializan tres líneas de producto: consolas de juego, software de entretenimiento y tabletas.

El simulador focaliza su actuación sobre el diseño, marketing y comercialización de los productos. Entre sus mecanismos de productividad está la orientación hacia la constante

capacitación del personal para elevar los indicadores de producción **[12]**.

Simulador BusinessGlobal reproduce el escenario competitivo de cinco compañías de tecnología del hogar que deben competir entre sí. Cada una de las compañías tiene una posición competitiva diferenciada, de acuerdo a su implantación en diferentes zonas geográficas a nivel internacional. Las compañías producen y comercializan tres líneas de producto: alta tecnología personal, pequeño electrodoméstico y domótica del hogar.

El simulador permite comprender las implicaciones del proceso de globalización sobre los diferentes eslabones de la cadena de valor (organización, financiación, innovación, producción, marketing). El simulador considera hasta 10 mercados y en cada uno de ellos 3 segmentos de demanda (precio, innovación y prestaciones) **[12]**.

Simulador Reto LabsagOp: el sitio Reto Labsag presenta una competencia de simulación de negocios que se realiza desde el año 2005. La participación en el Reto Labsag no tiene costo alguno para las universidades asociadas, solo es necesario formar uno o más equipos de alumnos con la tutoría de un profesor, todos pertenecientes al proceso productivo**[13]**.

Todas estas herramientas muestran el comportamiento de un escenario empresarial, tratando los conceptos de mantenimiento, productividad del personal y estado de la maquinaria; siendo capaz de incorporar conocimiento a los estudiantes y a su vez arrojar decisiones a tener en cuenta en el área de producción de una empresa productiva, sin embargo, todas son aplicaciones online, donde el usuario debe interactuar en tiempo real con las mismas. Además, son de licencia privativa, no permitiendo acceder así al código fuente para su posterior modificación, o identificar los métodos y técnicas empleadas para definir coeficientes de mantenimiento o productividad laboral, por citar solo estos. En la Tabla 1 se muestran los indicadores definidos para el análisis de los simuladores de negocios estudiados.

Simulador	Área Productiva	Factores Productivos	Aplicación de escritorio	Descripción del proceso productivo
Global2020	✓	✓	✗	✗
TechCompany	✓	✓	✗	✗
BusinessGlobal	✓	✗	✗	✗
LabsagOp	✓	✓	✗	✗

Tabla 1: Análisis de los indicadores en los simuladores estudiados.

Después de analizar los simuladores anteriores se pudo constatar que empleaban recursos matemáticos para estimar la productividad del personal y el coeficiente de disponibilidad técnica teniendo en cuenta los gastos empleados por concepto de mantenimiento y capacitación. Esta relación matemática no ha sido expuesta por los propietarios de los simuladores estudiados por lo que se realizó un estudio con mayor profundidad para determinar una función matemática capaz de estimar los indicadores planteados anteriormente.

1.4 Modelo matemático para el cálculo de los factores productivos.

Los factores productivos comprenden dos grupos: Recursos Materiales y Recursos Humanos, ambos se deben tener en cuenta para determinar la producción terminada de las empresas, siendo estos indicadores importantes a la hora realizar el modelo matemático de la investigación.

Para calcular la capacidad disponible(CD) es necesario identificar el efecto cuello de botella presente en las áreas de la línea de producción. Se necesita también conocer el Coeficiente de Disponibilidad Técnica(CDT) de la maquinaria, expresado de la siguiente manera: **[10]**

- *Bueno* $(0.9 \leq CDT \leq 1)$
- *Regular* $(0.5 \leq CDT < 0.9)$

- *Malo* $(CDT < 0.5)$

Para la presente investigación se considera que la inversión que realice la empresa en mantenimiento definirá cuán cercano a 1 estará el CDT, siendo la relación la que se define a continuación:

X	y
0,1	0,3
1	0,5
3	0,9
4	1

Tabla 2: Inversión en mantenimiento vs CDT

Donde la variable *x* se define como la inversión en mantenimiento en miles de pesos y la variable *y* es la capacidad disponible. Por tanto, una empresa que invierta mil pesos tendrá un CDT = 0.5, lo que significa que la línea de producción de su empresa estará trabajando al 50% de su capacidad instalada. El problema está dado en cómo estimar la CDT en aquellas empresas que inviertan una cantidad que se encuentre entre los valores definidos en la Tabla 2. Para ello se utilizará la **Interpolación Polinomial** que es un método para estimar aproximadamente, los valores que toma cierta función de la cual sólo se conoce su imagen en un número finito de abscisas. El objetivo será hallar un polinomio que cumpla lo antes mencionado y que permita hallar aproximaciones de otros valores desconocidos para la función con una precisión deseable fijada.

De manera similar ocurre con el indicador Aprovechamiento de la Jornada Laboral (AJL) expresado de la siguiente manera:

- *Bueno* $(0.85 \leq AJL \leq 1)$
- *Regular* $(0.4 \leq AJL < 0.85)$
- *Malo* $(AJL < 0.4)$

Para la presente investigación se considera que la inversión que realice la empresa en capacitación de personal definirá cuan cercano a 1 estará el AJL, siendo la relación la que se define a continuación:

x	y
0,8	0,2
1,8	0,6
3	1

Tabla 3: Inversión en capacitación vs AJL

Donde la variable *x* se define como la inversión en capacitación del personal en miles de pesos y la variable *y* es el valor de aprovechamiento laboral. Por tanto, una empresa que invierta tres mil pesos tendrá un AJL = 1, lo que significa que el personal tendrá un excelente aprovechamiento de la jornada laboral. El problema está dado en cómo estimar el AJL en aquellas empresas que inviertan una cantidad que se encuentre entre los valores definidos en la Tabla 3, para ello se emplea el método expresado anteriormente, adecuado a las características de este caso.

La jornada laboral en la presente investigación se considera de 8 horas de trabajo y 24 días al mes. El período estará definido por el profesor, puede tener la cualidad: trimestral, semestral o anual.

1.5 Metodología de desarrollo de software

Desarrollar el software adecuado depende de muchas actividades y etapas, donde el impacto de elegir la mejor metodología en un determinado proyecto es trascendental para el éxito del producto. El papel de las metodologías es sin duda esencial en un proyecto, esta debe adecuarse a las características del equipo, guiar y organizar las actividades que conlleven al cumplimiento de las metas trazadas en el grupo. Una metodología es una colección de métodos de solución de problemas organizados bajo una filosofía común y gobernados por un conjunto de principios.

Las metodologías se clasifican en dos grupos, las metodologías tradicionales o pesadas y las metodologías ágiles o ligeras. En el proceso de desarrollo de proyectos grandes y que

generan mucha documentación se recomienda emplear las metodologías tradicionales, mientras que para proyectos pequeños se recomiendan las metodologías ligeras **[15]**.

Las metodologías tradicionales centran su atención en llevar una documentación exhaustiva de todo el proceso de desarrollo del proyecto y en cumplir con la planificación definida en la fase inicial. Otra de las características importantes dentro de este enfoque son los altos costos al implementar un cambio y al no ofrecer una buena solución para proyectos donde el entorno es volátil **[14]**.

Las metodologías ágiles son usadas en proyectos cuyo objetivo fundamental es tener el software funcionando lo antes posible, de esta manera el cliente tendrá las primeras versiones donde podrá comprobar y aportar su idea de negocio. Con este objetivo, se trabaja sobre las versiones previas, siendo el desarrollo de la siguiente iteración una mejora de la anterior. Las metodologías ágiles proponen una forma de trabajo flexible cuya planificación se actualiza continuamente. Esto contrasta con las metodologías tradicionales, las cuales siguen una planificación precisa desde el principio **[15]**.

Debido a que se requiere reducir drásticamente el tiempo de desarrollo de la herramienta propuesta, que el equipo de trabajo es pequeño y existe un constante intercambio con el cliente, siendo este parte del equipo, se decide utilizar una metodología ágil. Dentro de las metodologías ágiles se destacan: AUP (Proceso Unificado Ágil), Scrum y XP (Programación Extrema)

AUP: se enfoca especialmente en la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Para ello, se crean y mantienen listas, identificando los riesgos desde etapas iniciales del proyecto. Especialmente relevante en este sentido es el desarrollo de prototipos ejecutables durante la base de elaboración del producto, donde se demuestre la validez de la arquitectura para los requisitos clave del producto y que determinan los riesgos técnicos **[16]**.

SCRUM: se centra en la gestión de proyectos y es óptima en aquellas situaciones en las que resulta complicado planificar el futuro con mecanismos de control de procesos empíricos. La retroalimentación entre iteraciones constituye el elemento más potente de la

metodología. El software es desarrollado por equipos que se auto organizan en iterativos e incrementales ciclos de corta duración (no más de 30 días) denominados *Sprints*, comenzando cada uno de ellos con una planificación y finalizando con una revisión retrospectiva. Como puede observarse, Scrum resulta válido en los entornos que trabajan con requisitos cambiantes, y necesitan rapidez de respuesta y flexibilidad **[17]**.

XP: según Pressman, XP es el enfoque más utilizado del desarrollo de software ágil. XP pone el énfasis en la colaboración estrecha pero informal (verbal) entre los clientes y los desarrolladores, en el establecimiento de metáforas para comunicar conceptos importantes, en la retroalimentación continua y en evitar la documentación voluminosa como medio de comunicación. Para alcanzar la simplicidad, XP restringe a los desarrolladores para que diseñen sólo para las necesidades inmediatas, en lugar de considerar las del futuro. El objetivo es crear un diseño sencillo que se implemente con facilidad en forma de código. La retroalimentación se obtiene de tres fuentes: el software implementado, el cliente y otros miembros del equipo de software **[18]**.

Se asume la metodología XP porque la propuesta de solución es un módulo para una herramienta informática existente y la metodología que se utiliza el equipo de desarrollo es precisamente esta.

1.5.1 Descripción de la metodología XP

El ciclo de vida de XP consiste básicamente de seis fases: Exploración, Planificación, Iteraciones por entregas, Producción, Mantenimiento y Muerte. A continuación se describe en que consiste cada fase **[19]**.

Exploración: en esta fase los clientes realizan las historias de usuario (HU) que desean que estén para la primera entrega. Cada HU describe una de las funcionalidades que el programa tendrá. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, la tecnología y las prácticas a ser utilizadas durante el proyecto. La duración de esta fase puede extenderse desde unas pocas semanas a varios meses dependiendo de la adaptación del equipo de desarrollo. En esta fase también se describen los requisitos funcionales y no funcionales de la aplicación.

Planificación: el objetivo de esta fase es fijar la prioridad de cada una de las HU y se establece cual será el contenido de la primera entrega. Los programadores estiman cuanto esfuerzo requiere cada HU y se establece el cronograma. El artefacto generado durante esta fase es la Estimación de esfuerzo por HU.

Iteraciones por entrega: esta fase incluye varias iteraciones del sistema antes de ser entregado. El calendario es dividido en un número de iteraciones de tal manera que cada iteración tome de una a cuatro semanas de implementación. En la primera iteración se crea un sistema que abarca los aspectos más importantes de la arquitectura global. El cliente decide que HU van a ser implementadas para cada iteración. Además, se realizan las pruebas funcionales, ejecutadas por el cliente, al final de cada iteración. Al final de la última iteración el sistema está listo para ser puesto en producción. Los artefactos generados en la misma son: el Plan de duración por iteración, las Tarjetas CRC (Clase, Responsabilidad y Colaborador) y las Tareas de ingeniería.

Producción: esta fase requiere realizar muchos más chequeos y pruebas al sistema antes de ser entregado al cliente. En esta fase aparecen nuevos cambios y se tiene que decidir si serán incorporados o no en dicha entrega. Durante esta fase suele suceder que las iteraciones se aceleren de tres a una semana. Las ideas pospuestas y las sugerencias son documentadas para luego ser implementadas más adelante. Los artefactos generados durante la fase son: el Acta de liberación del producto y el Acta de aceptación.

Mantenimiento: el sistema debe mantenerse en funcionamiento al mismo tiempo que desarrolle nuevas iteraciones. Esta fase puede requerir de nuevo personal o cambios en la estructura del equipo.

Muerte del proyecto: hay dos buenas razones por la cual el sistema entre en esta fase. La primera puede ser debido a que el cliente esté muy satisfecho con el sistema y no tenga ninguna otra funcionalidad que agregar en el futuro. La otra razón suele ser que el sistema no termina de ser liberado y el cliente necesita más funcionalidades y es imposible costearlas.

1.6 Herramientas de ingeniería del software asistida por computadora (CASE)

Las herramientas CASE ayudan a los gestores y practicantes de la ingeniería del software en todas las actividades asociadas a los procesos de software. Automatizan las actividades de gestión de proyectos, los productos de los trabajos elaborados a través del proceso, y ayudan a los ingenieros en el trabajo de análisis, diseño y codificación, permitiendo así obtener resultados adicionales y personalizados que no serán fáciles ni prácticos de producir sin el soporte de herramientas **[20]**.

Existen diversas herramientas CASE que soportan el lenguaje de modelado UML (Lenguaje Unificado de Modelado), entre estas se destacan Rational Rose y Visual Paradigm for UML. Ambas herramientas son muy potentes y permiten la generación de diversos diagramas como los de clases, de objetos, de casos de uso del negocio y de paquetes, además de generar código a partir de los mismos. Por ser multiplataforma, fácil de usar, proporcionar facilidad de trabajo con modelos UML y guardar todo el modelo en un solo fichero, se selecciona Visual Paradigm para UML en su versión 8.0 para la modelación del proceso de desarrollo de la herramienta propuesta. Además, se tuvo en cuenta que la universidad posee una licencia para su uso.

1.7 Lenguajes de programación

Un lenguaje de programación es un conjunto de reglas semánticas, así como sintácticas que los programadores usan para la codificación de instrucciones de un programa o algoritmo de programación. Es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo **[21]**.

Son diversos los lenguajes de programación existentes, debido a que se realiza un módulo para una herramienta informática de escritorio solo se analizaron los que se emplean para dichas aplicaciones. Según el ranking de los lenguajes de programación más populares de 2015 de la IEEE se destacan como más usados Java, C, C++, Python y C# **[22]**.

La herramienta existente se realizó con el lenguaje de programación Java, por lo que se decide continuar con su utilización para el desarrollo de la propuesta de solución. Dentro de las características de este lenguaje se encuentra:

- **Simple:** en Java hay un número reducido de formas claras para abordar una tarea dada. Ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de estas. Hereda la sintaxis de C/C++ y muchas de las características orientadas a objetos de C++ **[23]**.

- **Orientado a objetos:** se realiza una aproximación limpia, útil y pragmática a los objetos. El modelo de objeto de Java es simple y fácil de ampliar **[23]**.

- **Robusto:** Permite comprobar el código en tiempo de compilación y de ejecución. La liberación de memoria se realiza de forma automática, ya que se proporciona un recolector de basura automático para los objetos que no se utilizan. También proporciona una gestión de excepciones orientada a objetos **[23]**.

- **Arquitectura neutral:** Java se compila a un código de bytes de alto nivel independiente de la máquina. Este código está diseñado para ejecutarse en cualquier máquina con un sistema *runtime* (intérprete) el cual si es dependiente de esta **[23]**.

- **Portable:** implementa estándares de portabilidad, los enteros son siempre enteros, el sistema de interfaces de usuario lo constituye un sistema abstracto de ventanas, por lo que es independiente de la arquitectura en la que se implemente (UNIX, PC, Mac) **[23]**.

El estudio de las características del lenguaje de programación Java permitió profundizar en las facilidades que brinda el mismo, con el propósito de ser empleadas en el desarrollo del presente trabajo.

1.8 Entorno de desarrollo integrado

En el proceso de desarrollo emplear un IDE resulta ventajoso ya que este brinda muchas facilidades y beneficios, entre los cuales está conocer los ficheros en los que existe algún error de sintaxis.

Un Entorno de Desarrollo Integrado, traducido del inglés *Integrated Development Environment* (IDE) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Provee de un marco de trabajo amigable para la mayoría de los lenguajes de programación [**24**].

Existen diversos IDE entre los que sobresalen Netbeans y Eclipse. Se selecciona para el desarrollo de la solución informática la herramienta Netbeans en su versión 8.2, teniendo en cuenta que es libre, se integra perfectamente con Java que es el lenguaje de programación definido para el desarrollo de la solución propuesta, el equipo posee dominio de esta herramienta, además de un conjunto de elementos que se describen a continuación.

Netbeans es un entorno de desarrollo integrado, modular y de base estándar. Consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general para compilar cualquier tipo de solución. Presenta una interfaz amigable e intuitiva y tiene todas las herramientas para crear soluciones profesionales ya sean de escritorio, empresariales, web, móviles y aplicaciones SOA (Arquitectura Orientada a Servicios), no solo en Java sino también en C/C++ y Ruby. NetBeans ya viene con plugins y módulos integrados, evitando tener que configurar el ambiente, brindando todo el entorno listo para trabajar [**24**].

1.9 Sistema gestor de bases de datos

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos. También ayuda al mantenimiento de la integridad de los datos y controla la

seguridad y privacidad **[25]**. Existen variados sistemas gestores de bases de datos entre ellos se encuentran MySQL, PostgreSQL, Apache Derby y Oracle.

SQLite es una biblioteca en proceso que implementa un sistema autónomo, sin servidor, sin necesidad de configuración, el motor transaccional de la base de datos SQL. El código de SQLite es de dominio público y por lo tanto libre para el uso para cualquier propósito, comercial o privado. SQLite es una base de datos SQL embebido motor. A diferencia de la mayoría de otras bases de datos SQL, esta no tiene un proceso servidor independiente, este lee y escribe directamente a los archivos de disco normal. Una completa base de datos SQL con varias tablas, índices, triggers y vistas, está contenida en un archivo de disco único. El formato de archivo de base de datos es multi-plataforma que libremente puede copiar una base de datos entre sistemas de 32-bit y 64-bit. Estas características hacen que SQLite una opción popular como un formato de archivo de aplicación, siendo además de gran utilidad para la aplicación en desarrollo, la misma se puede manipular desde cualquiera de los gestores mencionados anteriormente[**25**].

1.10 Patrones de diseño

Los patrones de diseño son descripciones de la comunicación de objetos y clases que pueden personalizarse para resolver un problema de diseño en general en un contexto particular [**27]**.

Los patrones de diseño son soluciones para problemas típicos y recurrentes que pueden aparecer durante el desarrollo de una aplicación, estos ayudan a estandarizar el código y a hacer que el diseño sea más comprensible para otros programadores. Se pueden agrupar en dos grandes grupos; los GRASP (del inglés *General Responsibility Assignment Software Patterns*), que son patrones generales de software para asignación de responsabilidades y los GOF (*Gang of Four, en inglés*), encargados de la inicialización, agrupación y comunicación de los objetos [**27**].

En el capítulo 2 se muestran los patrones de diseño utilizados en la solución informática.

1.11 Pruebas

El desarrollo de los sistemas de software implica una serie de actividades de producción en las que las posibilidades que el fallo humano aparezca son grandes. Los errores pueden empezar a darse desde el primer momento del proceso, en el que los objetivos pueden estar especificados de forma errónea, debido a la imposibilidad del hombre de trabajar y comunicarse de forma perfecta. El desarrollo de software debe ir acompañado de una actividad que garantice la calidad, es por esto que las pruebas de software constituyen un elemento crítico para la garantía de la calidad del software. Estas pruebas se ejecutan dirigidas a componentes o al sistema en su totalidad, con el objetivo de medir el grado en el que este cumple con los requisitos pactados. De los niveles de pruebas existentes se realizarán los siguientes:

Pruebas de unidad: se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño, ya sea un componente o un módulo del software **[28]**.

Pruebas de sistema: abarca una serie de pruebas diferentes cuyo propósito principal es ejercitar profundamente el sistema de cómputo. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se hayan integrado adecuadamente todos los elementos del sistema y que realizan las funciones adecuadas **[28]**.

Pruebas de aceptación: una vez culminado el proceso de pruebas por parte del equipo de desarrollo, es indispensable, que el cliente verifique que el producto ha sido desarrollado con las normas y criterios establecidos, y cumple con todos los requisitos especificados por el cliente **[29]**.

1.12 Conclusiones parciales

El análisis de la simulación en la actualidad, el aporte de los simuladores al proceso docente acercando a los estudiantes a la gestión empresarial, la utilización de los factores productivos en la producción terminada, constituyen la base teórica de la solución propuesta.

El estudio de las soluciones existentes para el cálculo de la producción terminada, permitió identificar los indicadores empleados para el cálculo de la producción terminada y aplicarlos en el desarrollo de la solución propuesta. El estudio de las metodologías y tecnologías utilizadas en el desarrollo de software, permitió ampliar los conocimientos para su posterior uso.

CAPÍTULO 2: ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA INFORMÁTICA

2.1 Introducción

En el capítulo se exponen las principales características del sistema desarrollado, mediante el modelado del negocio y la identificación de los requisitos funcionales y no funcionales. Además, se describe la arquitectura por la cual se rigió la solución obtenida y los patrones de diseño que fueron utilizados para dar solución al problema que dio origen a la presente investigación.

2.2 Descripción del sistema

La herramienta informática propuesta permitirá determinar la producción terminada teniendo en cuenta los factores productivos: recursos materiales, humanos y tecnológicos. Se analiza además la capacidad disponible de la línea de producción, dado por el indicador CDT, así como la productividad del personal expresado por el indicador de AJL.

Cada equipo debe tomar diferentes decisiones en cuanto a la gestión de su empresa virtual, teniendo en cuenta el escenario brindado por el profesor de la asignatura. A través de las decisiones los estudiantes tendrán la oportunidad de reorientar la estrategia y táctica a seguir. Entre las decisiones que pueden tomar se encuentran:

- Volumen de producción: Cada período se debe decidir cuantas unidades de producto a realizar.

- Volumen de materia prima: En cada período se debe decidir las unidades de materia prima que se desea comprar. El consumo de materia prima por producto estará definido por la cantidad de piezas necesaria para la confección de un producto, definidas previamente por el profesor. La materia prima ordenada en un período ya está disponible para la producción en el inicio del período que se trate.

• Adquisición de nuevas líneas de producción. Al inicio de la simulación los equipos cuentan con una línea de producción, pero pueden decir instalar nuevas líneas. Solo se contemplan líneas de producción para grandes confecciones de productos.

• Desmantelar líneas de producción: Se pueden vender las líneas de producción en cualquier período. Una línea de producción que ha sido vendida ya no está disponible al comienzo del período que se trate.

• Inversión en Mantenimiento: Las líneas de producción están sujetas a constantes desgastes. Las reducciones en la capacidad que pueden resultar del uso y desgaste de estas pueden evitarse o mantenerse bajo control, asegurando que las partes de la planta en necesidad de reparación se les proporcione mantenimiento con regularidad.

• Inversión en capacitación del personal: El aumento en la inversión de la capacitación del personal de producción mejora las habilidades de los empleados y conduce a un mayor índice de productividad.

En la Figura 1, se presenta el proceso de familiarización con el simulador y las fases por la que transita el mismo.

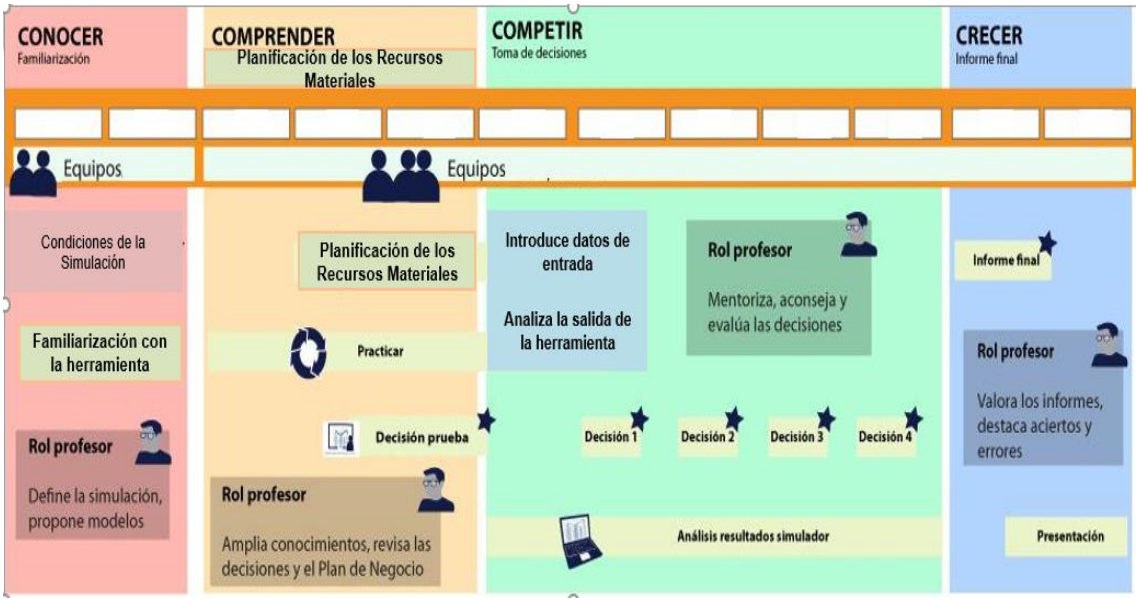

Figura 1: Proceso de familiarización con el simulador

En la fase de **conocer** el profesor establece las condiciones iniciales de la simulación, es decir, se encarga de crear una nueva línea de producción con sus características expresadas en la Figura 12: Modelo de datos del sistema.

Por otra parte, el estudiante se familiariza con la herramienta, sus funcionalidades y manera de utilizar.

En la fase de **comprender** el profesor revisa las decisiones formuladas y el Plan de Recursos Materiales creados por los estudiantes; estos a su vez, realizan pruebas iniciales para practicar sobre la herramienta.

En la fase **competir** el profesor se encarga de monitorizar y aconsejar a los estudiantes sobre las decisiones tomadas. Por su parte el simulador procesa los datos introducidos por los estudiantes, para determinar la producción terminada de la empresa expresado en la Figura 2:

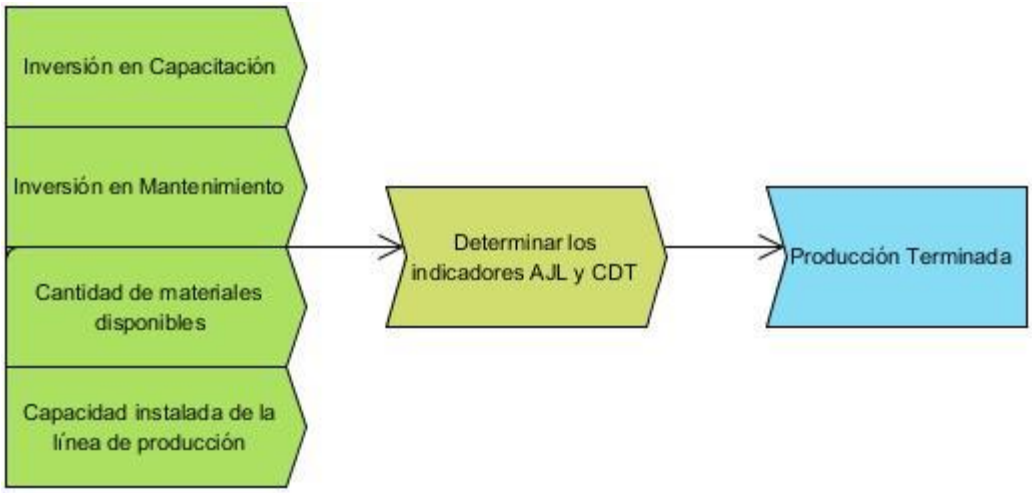

Figura 2: Determinación de la producción terminada

Los indicadores de AJT y CDT se determinan mediante la interpolación polinomial expuestos en el epígrafe 1.4. A continuación se muestran las funciones resultantes en cada caso.

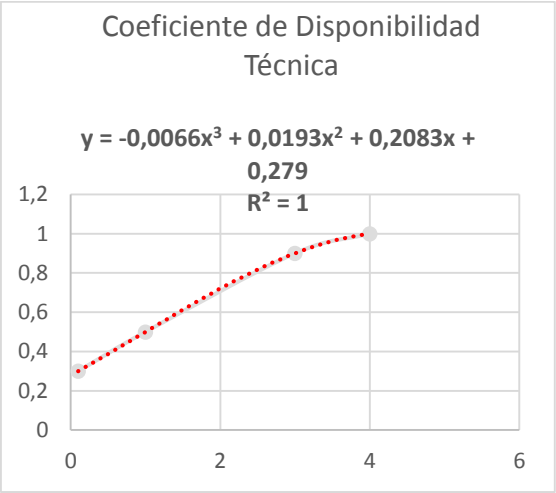

Figura 3: Análisis del indicador CDT

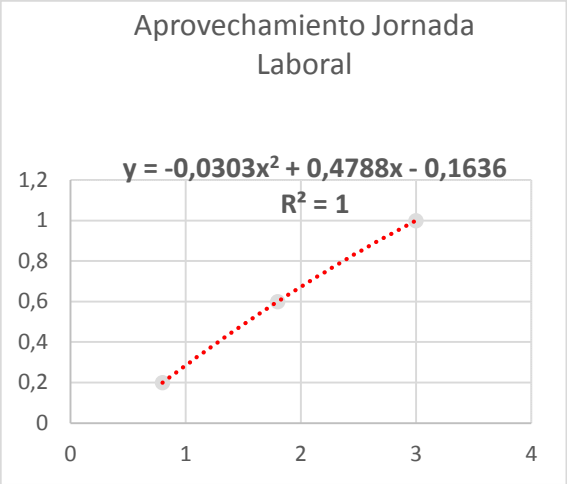

Figura 4: Análisis del indicador AJL

En ambas gráficas el coeficiente de determinación $\boldsymbol{R^2 = 1}$ esto significa que la línea de tendencia permite pronosticar valores para CDT y AJL con alto grado de certidumbre. Estos indicadores son utilizados posteriormente para el cálculo de la producción según los factores productivos: tecnología y recursos humanos respectivamente. A continuación, se definen las variables utilizadas en el modelo matemático:

PG: Producción en el período según el factor productivo **tecnología** medida en unidades.

CI: Capacidad instalada de las líneas de producción en unidad de producción en el período.

PH: Producción en el período según el factor productivo **recursos humanos** medida en unidades.

PT: Producción terminada en el período medida en unidades.

JL: Jornada Laboral definida como minutos por período, considerándose en la investigación que se trabajan 8 horas diarias durante 24 días al mes.

CT: Cantidad de trabajadores en el período.

PE: Período de tiempo en que se realiza la simulación, siendo un mes la unidad mínima de tiempo por tanto **PE ≥1.**

TP: tiempo promedio que se emplea en realizar un producto medido en minutos por unidades de producto.

PM: Producción en el período según el factor productivo **recursos materiales** medida en unidades.

PI: Número de piezas necesarias para la confección de un producto medidas en piezas por producto.

MP: Volumen de materia prima destinada a la producción medida en piezas por período.

Las relaciones funcionales se definen de la siguiente forma:

$$\boldsymbol{PG = CI * CDT} \qquad \boldsymbol{JL = 24 * 8 * 60} \qquad \boldsymbol{PH = \left(\frac{JL}{TP}\right) * AJL * CT * PE} \qquad \boldsymbol{PM = MP \div PI}$$

Una vez definido por el sistema los valores antes expuestos, el sistema decide la producción terminada de la empresa, calculado de la siguiente manera:

IF (($\boldsymbol{PM \leq PH}$) && ($\boldsymbol{PM \leq PG}$))

THEN $\boldsymbol{PT = PM}$;

ELSE, IF(($\boldsymbol{PM > PH}$) && ($\boldsymbol{PM > PG}$) && ($\boldsymbol{PH > PG}$))

THEN $\boldsymbol{PT = PG}$;

ELSE $\boldsymbol{PT = PH}$;

En la fase **crecer** el estudiante analiza el informe brindado por el sistema sobre la situación productiva de la empresa. Mientras, el profesor valora estos informes y destaca los aciertos y corrige los errores sobre las decisiones tomadas.

2.3 Fases del proceso de desarrollo

Teniendo en cuenta que en el capítulo 1 se define la metodología XP para guiar el proceso de desarrollo de la presente investigación. A continuación, se describe como fueron aplicadas las fases de exploración, planificación e iteraciones por entrega y los artefactos generados en cada una de ellas. La fase producción será descrita en el siguiente capítulo.

2.3.1 Fase de exploración

En esta fase se describen los requisitos de la herramienta a través de las HU. También el equipo de desarrollo se familiariza con las tecnologías y herramientas a utilizar en el desarrollo del proyecto.

2.3.1.1 Requisitos de la herramienta

Los requisitos cumplen un papel primordial en el proceso de producción de software, su principal tarea consiste en la generación de especificaciones correctas que describan con claridad el comportamiento del sistema para así minimizar los problemas relacionados al desarrollo de sistemas. Estos se enfocan en la definición de lo que se desea producir y tienen dos clasificaciones: requisitos funcionales y no funcionales. Los requisitos funcionales indican lo que debe hacer la solución propuesta, es decir, lo que el sistema hace para el usuario. Los requisitos no funcionales, por su parte, se refieren a propiedades que debe cumplir el software, tales como capacidad de almacenamiento, fiabilidad o mantenibilidad. A continuación, se listan los requisitos de la herramienta propuesta.

Requisitos funcionales

RF1: Crear Línea de Producción

RF2: Mostrar Línea de Producción

RF3: Eliminar Línea de Producción

RF4: Determinar el Coeficiente de Disponibilidad Técnica

RF5: Determinar el Aprovechamiento de la Jornada Laboral

RF6: Determinar Producción Terminada

Requisitos no funcionales

Requisitos de usabilidad

RNF 1: La herramienta tendrá visible la opción de Ayuda.

RNF 2: La herramienta debe proporcionar mensajes que sean informativos y orientados al usuario final.

RNF 3: La herramienta debe contar con una interfaz amigable y de fácil entendimiento.

RNF 4: Emplear el mismo formato de letra en toda la herramienta.

Requisito de software

RNF 5: La herramienta es multiplataforma.

Requisitos de hardware

RNF 6: Para ejecutar la herramienta la computadora debe contar con las siguientes características:

- 512 MB de memoria RAM.
- Procesador a 3.2 Ghz, equivalente o superior.
- Capacidad de 1 GB de disco duro.

Requisito de seguridad

RNF 7: El sistema debe garantizar el control de acceso a través de la autenticación de usuarios para la administración del mismo.

2.3.1.2 Historias de usuario

La HU es la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas **[30]**. A continuación, se describe la HU de prioridad alta en el negocio. El resto de las HU se exponen en los anexos.

Historia de Usuario	
Número: 4	**Nombre de la Historia de Usuario:** Determinar Producción Terminada
Modificación a la Historia de Usuario: Ninguna	
Usuario: Osiel Sánchez Martínez	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 3 semanas
Riesgo en desarrollo: Bajo	**Puntos reales:** 3 semanas
Programador responsable: Osiel Sánchez Martínez	
Descripción: **Determinar Producción Terminada:** permite determinar la producción terminada de la empresa en una etapa determinada.	
Observaciones: **Determinar Producción Terminada**: Anteriormente el usuario debe determinar los indicadores: CDT y AJL.	

Tabla 4:HU "Determinar Producción Terminada".

2.3.2 Fase de planificación

En esta fase el cliente establece la prioridad de cada historia de usuario y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se

toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Esta fase dura unos pocos días [**30**].

2.3.2.1 Estimación de esfuerzo por historias de usuario

La estimación de esfuerzo asociado a la implementación de las historias de usuario la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Si la estimación supera las 3 semanas, la HU deberá ser dividida hasta que pueda ser desarrollada en un tiempo factible. En caso de que el esfuerzo sea menor que una semana, la HU será combinada por otra [**30**]. En la siguiente tabla se muestra la estimación de esfuerzos por HU para el desarrollo de la herramienta propuesta:

Historias de usuario	Puntos de estimación
Gestionar Líneas de Producción	1
Determinar Coeficiente de Disponibilidad Técnica	1
Determinar Coeficiente de Aprovechamiento de la Jornada Laboral	1
Determinar Producción Terminada	3

Tabla 5: "Estimación de esfuerzo por HU".

2.3.3 Fase de iteraciones por entrega

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última iteración el sistema estará listo para entrar en producción [**30**]. A continuación, se describen cada una de las iteraciones propuestas.

Iteración #1: en esta iteración se implementaron las HU 1, las cuales se refieren a la gestión de las líneas de producción, dígase de los RF crear línea de producción, mostrar línea de producción y eliminar línea de producción.

Iteración #2: en esta iteración se implementaron las HU 2, 3 las cuales se refieren a los requisitos funcionales Determinar Coeficiente de Disponibilidad Técnica y Determinar Coeficiente de Aprovechamiento de la Jornada Laboral.

Iteración #3: en esta iteración se implementó las HU 4 la cual se refiere al requisito funcional Determinar Producción Terminada.

2.4 Diseño de la herramienta

La metodología de desarrollo de software XP propone que se realicen diseños simples y sencillos, que permita un fácil entendimiento a los desarrolladores, permitiendo reducir el tiempo al realizar las tareas de implementación.

2.4.1 Arquitectura

La arquitectura de software representa estructuras del sistema que consiste en componentes de software, las propiedades externas visibles de esos componentes y las relaciones entre ellos [32]. Para el desarrollo de la presente investigación se utilizó el patrón arquitectónico Modelo- Vista- Controlador (MVC), el cual tiene como objetivo principal separar el código responsable de la representación de los datos del código encargado de almacenarlos. Para lograrlo, la arquitectura divide la capa de presentación en tres tipos de objetos básicos; que son: modelos, vistas y controladores como se muestra en la Figura 5. La utilidad de esta reside en que describe los flujos de comunicación entre los tres tipos de objetos.

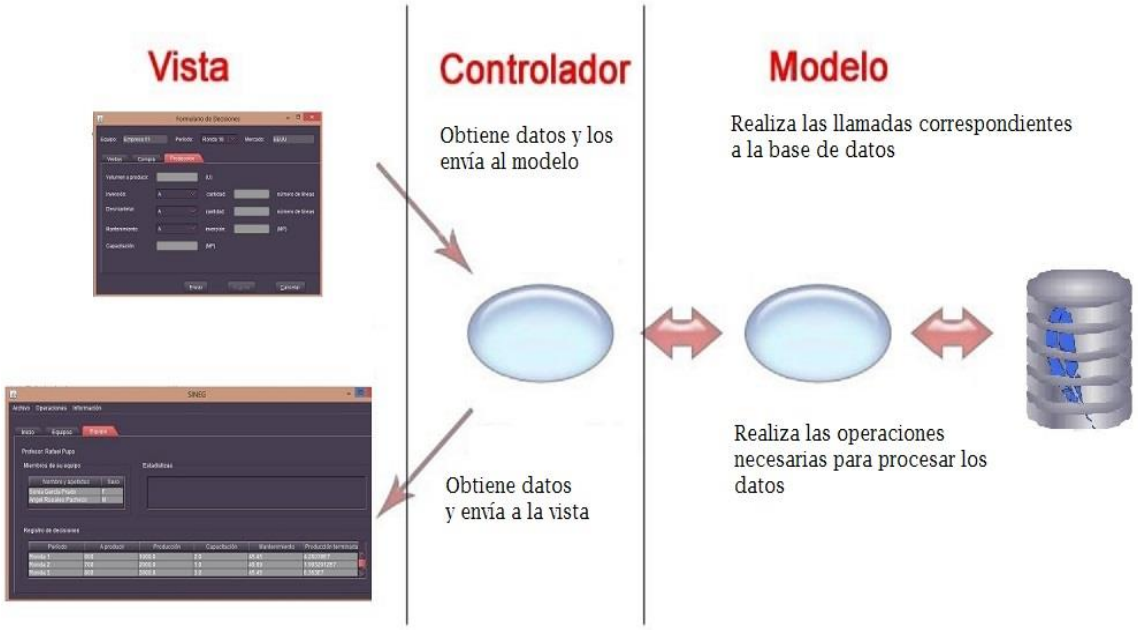

Figura 5: Arquitectura Modelo Vista Controlador

En un escenario clásico, la vista constituye la representación visual de la interfaz de usuario y el modelo representa la estructura lógica, independientemente de su representación visual. El controlador recoge las acciones del usuario, interactúa con el modelo y devuelve una determinada vista en respuesta. En la mayor parte de las implementaciones de MVC, los tres componentes pueden relacionarse directamente el uno con el otro, pero en algunas implementaciones el controlador es responsable de determinar qué vista mostrar. Esta arquitectura permitirá reducir la complejidad en el diseño arquitectónico e incrementar la flexibilidad y mantenimiento del código [**32**].

Este patrón de software permite implementar cada uno de los componentes por separado, donde la vista y el controlador dependen de las interacciones del modelo. La conexión entre el modelo y sus vistas es dinámica, realizando el intercambio de información en tiempo de ejecución. La forma en que el patrón se estructura simplifica la comprensión y la organización del desarrollo del sistema, reduciendo las dependencias [**32**].

El patrón MVC fue diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales están dadas por el hecho de que, el modelo, las vistas y los

controladores se tratan como entidades separadas; lo que hace que cualquier cambio producido en el modelo se refleje automáticamente en cada una de las vistas. Este modelo de arquitectura se puede emplear en sistemas de representación gráfica de datos, donde se presentan partes del diseño con diferente escala de aumento, en ventanas separadas.

Este modelo de arquitectura presenta las siguientes ventajas:

- ✓ Separación clara entre los componentes de un programa; lo cual permite su implementación por separado.
- ✓ Interfaz de Programación de Aplicaciones (API) bien definida; cualquiera que use las API, podrá reemplazar el modelo, la vista o el controlador sin aparente dificultad.
- ✓ Conexión dinámica entre el modelo y sus vistas; se produce en tiempo de ejecución, no en tiempo de compilación.

A continuación, se muestra la distribución de las clases empleadas en la programación y su correspondencia con las capas de la arquitectura.

Figura 6: Clases de la capa modelo de datos.

En la capa del modelo se encuentran las clases destinadas a la manipulación de datos. Las clases Conector.java y Conexión.java se encargan de la conexión a la base de datos, a su vez, las clases Configuración.java y CondicionInicial.java se encargan de la configuración de los datos ofrecidos por el usuario que posteriormente son manipulados e introducidos a la base de datos. Se encuentra también, Produccion.java como clase capaz de gestionar los datos asociados a la producción de la empresa y, por último, se cuenta con la clase

Decisiones.java para almacenar las correspondientes decisiones a tener en cuenta durante el proceso.

Figura 7: Clases de la capa vista

En la capa de las vistas se encuentran las clases encargadas de mostrar, mediante las interfaces, los datos al usuario. Dígase las vistas destinadas a configuración inicial, las líneas de producción, las decisiones y el control de usuario.

Figura 8: Clases de la capa controladora

En la capa Controlador se encuentra la clase Controladora.java, la cual es la encargada de gestionar las peticiones realizadas para manipular los datos y mostrarlos mediante las interfaces.

2.4.2 Patrones de diseño

Para diseñar la herramienta se empleó alguno de los patrones de diseño, siendo estos la base para la búsqueda de soluciones a problemas reales en el desarrollo de software. A continuación, se describen los patrones empleados:

Patrones de diseño GRASP

- **Experto:** se pone en práctica con el uso de clases que poseen responsabilidades específicas a cumplir, de acuerdo con la información que manejan. El uso de este patrón se evidencia en la clase *Juego.java* puesto que se especializa en la gestión de las líneas de producción. Un ejemplo es el método *ComprarLineas* expuesto en el siguiente fragmento de código.

```
public void ComprarLineas(int num_equipo, LineaProduccion nueva, int cant)throws Exception{
 double costo = nueva.getPrecio() * cant;
 double ex = 0;

    for (Equipo equipo : equipos) {
        if (equipo.numero_equipo == num_equipo) {
            if (equipo.presupuesto_inicial >= costo) {
                while (cant != 0) {
                    equipo.getLineas().add(nueva);
                    cant --;
                }
                ex = equipo.getPresupuesto_inicial() - costo;
                equipo.setPresupuesto_inicial(ex);
                throw new Exception(" Su compra se ha realizado con éxito");
            }
            throw new Exception(" Su empresa no cuenta con el dinero suficiente para realizar esta operación");
        }
        throw new Exception(" El número del equipo es incorrecto");
    } }
```

Figura 9: Ejemplo del patrón de diseño experto

- **Creador:** se refleja en las clases que tienen la responsabilidad de instanciar objetos de otras clases. El uso de este patrón se evidencia en la clase *Decision.java* al ser la encarga de crear la decisión de crear una nueva línea de producción.

```java
public void ComprarLineas(int num_equipo, LineaProduccion nueva, int cant)throws Exception{
 double costo = nueva.getPrecio() * cant;
 double ex = 0;

    for (Equipo equipo : equipos) {
        if (equipo.numero_equipo == num_equipo) {
            if (equipo.presupuesto_inicial >= costo) {
                while (cant != 0) {
                    equipo.getLineas().add(nueva);
                    cant --;
                }
                ex = equipo.getPresupuesto_inicial() - costo;
                equipo.setPresupuesto_inicial(ex);
                throw new Exception(" Su compra se ha realizado con éxito");
            }
            throw new Exception(" Su empresa no cuenta con el dinero suficiente para realizar esta operación");
        }
        throw new Exception(" El número del equipo es incorrecto");
    }
```

Figura 10: Ejemplo del patrón de diseño creador

Controlador: es el experto coordinando el trabajo de otros expertos. Este patrón se evidencia en la clase *Controladora.java* que es la encargada de la gestión productiva de la empresa.

```
public class Controladora {

    private Conector conexion;
    Statement state = null;
    PreparedStatement prepare = null;
    ResultSet result = null;
    //private CondicionInicial cond_ini;
    private String usuario;
    private boolean privilegios;

    public Controladora() {
        conexion = new Conector();
        usuario = "";
        privilegios = false;
    //    cond_ini = conexion.ObtenerCondicionInicial();
    }
```

Figura 11: Ejemplo del patrón de diseño controlador

- **Alta cohesión:** se aplica para realizar un diseño que evite contener clases con un alto grado de abstracción, que asuman responsabilidades que podían haber delegado a otros objetos o que tengan responsabilidades complejas. El patrón se evidencia en cada de una de las clases de la herramienta, de tal forma que se elimina la sobrecarga de responsabilidades.

- **Bajo acoplamiento:** el acoplamiento mide la fuerza con que una clase está conectada a otra, de esta forma una clase con bajo acoplamiento debe tener un número mínimo de dependencia con otras clases. Este patrón se tuvo presente debido a la importancia que se le atribuye a realizar un diseño de clases independientes que puedan soportar los cambios de una manera fácil y que a su vez permitan la reutilización. El patrón se evidencia en cada una de las clases diseñadas para la herramienta.

Patrones de diseño GOF

- Instancia única (Singleton): presenta un mecanismo para limitar el número de instancias de una clase. Su uso se aprecia en las clases *GestorLinea,* proporcionando una instancia única a la clase PrincipalController de la capa de Presentación, existiendo así, un solo objeto de las clases encargadas de la gestión del negocio.

2.4.3 Estándares de codificación

El estándar de codificación empleado en el desarrollo de la aplicación fue definido por el equipo de desarrollo. A continuación, se muestran algunas pautas empleadas en dicho estándar.

- Todas las nomenclaturas a utilizar se definirán en idioma español.
- Los nombres de los paquetes y las clases serán con mayúscula, en caso de ser un nombre compuesto las palabras siguientes se escribirán de igual manera.
- Los nombres de los métodos serán con mayúscula, en caso de ser un nombre compuesto las palabras siguientes se escribirán con letra inicial mayúscula.
- Los identificadores para las variables y los parámetros serán con letras minúsculas y en caso de ser un nombre compuesto las palabras siguientes se escribirán con letra inicial mayúscula.
- Las clases de configuración de negocio comienza con el prefijo Config y luego el nombre de la clase (ConfigProduccion.java).
- Los nombres de variables o funciones deben ser lo suficientemente descriptivos, sin exceder de 30 caracteres.

- Todas las funciones deben tener comentarios explicando que realiza cada una de ellas.

2.5 Descripción de la base de datos

El modelo de datos permite estructurar la base de datos, en cuanto a los tipos de datos presentes y la forma en que se relacionan entre sí **[33]**. El siguiente modelo representa las 8 entidades con que cuenta la aplicación a la cual se integra la solución propuesta para el almacenamiento de toda la información. Se evidencia la utilización del patrón llave subrogada en sus entidades.

Al proceso de organizar los datos de una base de datos se le conoce como normalización, en el cual se incluye la creación de tablas y el establecimiento de relaciones entre ellas según reglas diseñadas tanto para proteger los datos como para hacer que la base de datos sea más flexible al eliminar la redundancia y las dependencias incoherentes. En una base

de datos normalizada lo habitual es que exista un mayor número de tablas que en otra que no lo está. En consecuencia, las tablas normalizadas suelen ser más pequeñas y con menos atributos que las tablas no normalizadas **[34]**.

A continuación, se describen brevemente las 6 formas de normalización para las bases de datos, partiendo desde la primera forma hasta la tercera, incluyendo la forma especial de Boyce-Codd y las dos últimas formas.

1. **Primera forma normal:** un campo dado de un registro dado, solo puede contener un valor, para evitar la redundancia de grupos dentro de un único registro.
2. **Segunda forma normal:** debe cumplir con las restricciones de la primera forma normal, así como cada campo no clave debe depender de la clave principal de la tabla. Además, dos o más tablas no pueden tener la misma clave principal, pues en caso de que esto suceda, para normalizar se combinan dichas tablas especializándolas en una sola, con una única clave principal.
3. **Tercera forma normal:** debe cumplir con las restricciones de la segunda forma normal y no deben existir dependencias funcionales transitivas.
4. **Forma normal de Boyce-Codd:** debe cumplir con las restricciones de la tercera forma normal y cada atributo determinante debe ser una clave candidata.
5. **Cuarta forma normal:** debe cumplir con todas las restricciones de la 3era forma normal y no debe tener dependencias multievaluadas.
6. **Quinta forma normal:** debe cumplir con las restricciones de la cuarta forma normal y cada entidad debe supeditar sus dependencias con las claves candidatas.

A partir del estudio documentado previamente, se puede concluir que la base de datos para la propuesta de solución se encuentra en 3FN. Ello implica que no existan relaciones demasiado complejas entre tablas, que provoquen dificultades a la hora de recuperar información en forma de consultas, difíciles de expresar por parte del programador o el usuario, eliminando de esta manera la dependencia transitiva **[34]**.

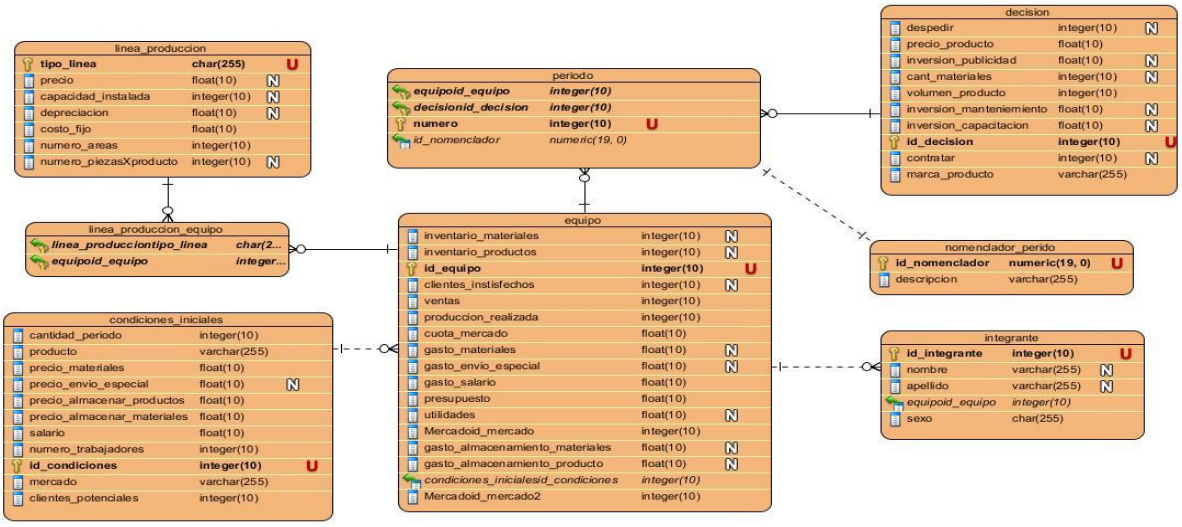

Figura 12: Modelo de datos del sistema.

A continuación, se describen las entidades relacionadas con la aplicación propuesta:

linea_produccion: Cada línea de producción contiene su identificador (***tipo_linea)***, así como los valores: *precio*, *costo fijo*, *número de área*, *número de piezas por producto*, así como la *depreciación* y la *capacidad instalada.*

equipo: Cada equipo tiene un identificador (***id_equipo***), así como el valor asociado a la *producción realizada.*

decision: Cada decisión contiene su identificador(***id_decision***), además de la relación de decisiones que son tomadas por el equipo; dígase: cantidad de materiales, volumen de productos, inversión en mantenimiento e inversión en capacitación.

2.6 Conclusiones parciales

- El empleo de la metodología XP en función de analizar el proceso de desarrollo de la herramienta permitió realizar un trabajo estructurado, generando los artefactos en cada una de sus fases, en correspondencia con el cronograma de desarrollo propuesto.
- El uso de una arquitectura MVC, así como el empleo de patrones de diseño, garantizaron la obtención de una solución con poca dependencia entre clases, flexible al mantenimiento y la aceptación de cambios.

CAPÍTULO 3: PRUEBAS DE LA HERRAMIENTA INFORMÁTICA

3.1 Introducción

En el presente capítulo se describe la fase producción de la metodología de desarrollo empleada, en la cual se le realizaron chequeos y pruebas a la herramienta desarrollada antes de ser entregada. Se exponen además los artefactos obtenidos, tales como: el acta

de liberación y de aceptación. Se realiza una validación de los resultados de la herramienta en relación al cumplimiento del objetivo general de la investigación.

3.2 Técnica de validación de los requisitos

Con el objetivo de validar que los requisitos previamente definidos cumplen con las expectativas del cliente, se aplicaron las técnicas de validación de requisitos:

- **Revisiones formales de los requisitos:** se realizaron revisiones formales a cada uno de los requisitos por parte del cliente y del equipo de desarrollo, obteniéndose un total de 4 no conformidades de tipo técnica, de redacción y formato, las que fueron corregidas en tiempo.

- **Construcción de prototipos:** se confeccionaron prototipos no funcionales, dando la posibilidad al cliente de poder comprobar de forma visual cómo quedaría la herramienta, obteniéndose finalmente un prototipo que satisface al cliente.

3.3 Validación del diseño

Un aspecto importante a tener en cuenta en la evaluación del diseño, es la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objetos. Para validar el diseño realizado se tendrán en cuenta distintos atributos de calidad como: la responsabilidad de las clases, la reutilización, la complejidad de implementación y mantenimiento, el bajo acoplamiento y la cantidad de pruebas unitarias. Las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC) permiten evaluar estos atributos **[18].**

3.3.1 Relaciones entre clases

La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso es evaluar un conjunto de atributos de calidad entre los que se encuentran el acoplamiento, complejidad de mantenimiento y reutilización de cada clase **[18]**.

A continuación, se explican los pasos que se llevaron a cabo para aplicar la métrica:

1. Determinar la cantidad de relaciones de uso (CRU) que poseen las clases a medir.

2. Calcular el promedio de las CRU.

3. Teniendo en cuenta los valores antes obtenidos determinar la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la tabla siguiente.

Atributos de calidad	Clasificación	Criterio
Acoplamiento	Ninguna	CRU = 0
	Baja	CRU = 1
	Media	CRU = 2
	Alta	CRU > 2
Complejidad de mantenimiento	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio
Reutilización	Baja	CRU > 2* promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU <= Promedio
Cantidad de pruebas	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio

Tabla 6: Criterios de evaluación de la métrica RC

Resultados del instrumento de evaluación de la métrica Relaciones entre Clases

Acoplamiento	Cantidad de Clases	Promedio
Ninguno	0	0
Bajo	6	66.66
Medio	3	33.33
Alto	0	0

Tabla 7: Incidencia de los resultados de la evaluación de la métrica RC en el atributo acoplamiento

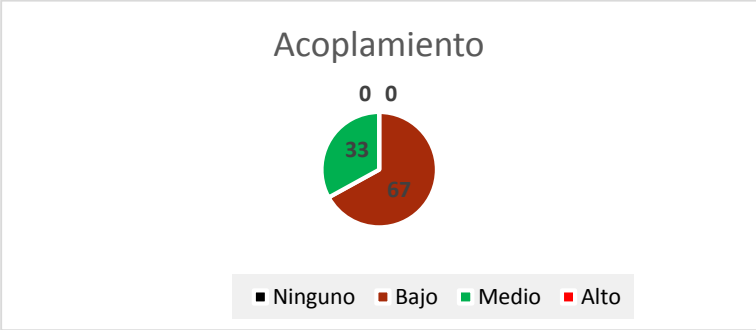

Figura 13: Representación gráfica de la incidencia de los resultados de la evaluación de la métrica RC en el atributo acoplamiento

Complejidad de Mantenimiento	Cantidad de Clases	Promedio
Bajo	9	100
Medio	0	0
Alto	0	0

Tabla 8: Incidencia de los resultados de la evaluación de la métrica RC en el atributo complejidad de mantenimiento

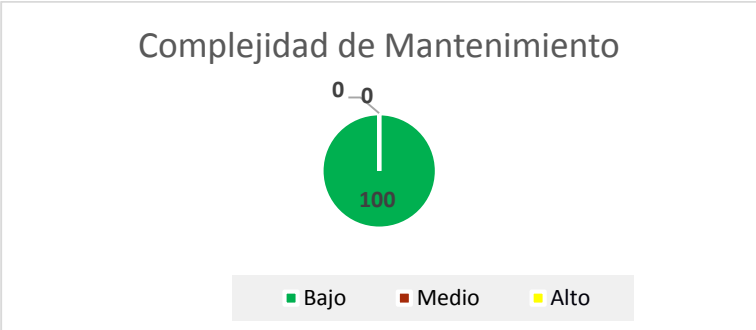

Figura 14: Representación gráfica de la incidencia de los resultados de la evaluación de la métrica RC en el atributo complejidad de mantenimiento

Reutilización	Cantidad de Clases	Promedio
Bajo	0	0
Medio	1	11.2
Alto	8	88.8

Tabla 9: Incidencia de los resultados de la evaluación de la métrica RC en el atributo reutilización

Figura 15: Representación gráfica de la incidencia de los resultados de la evaluación de la métrica RC en el atributo reutilización

Cantidad de Pruebas	Cantidad de Clases	Promedio
Bajo	8	88.8
Medio	1	11.2
Alto	0	0

Tabla 10: Incidencia de los resultados de la evaluación de la métrica RC en el atributo cantidad de pruebas

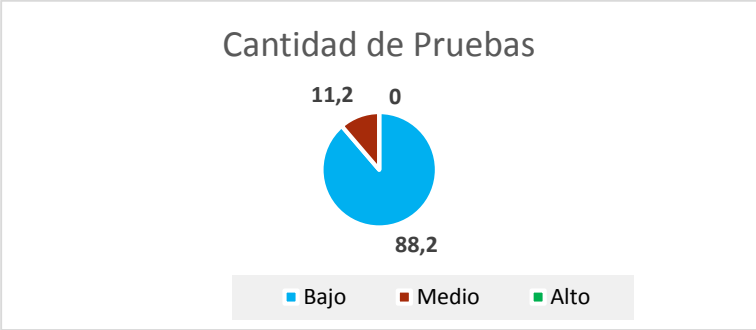

Figura 16: Representación gráfica de la incidencia de los resultados de la evaluación de la métrica RC en el atributo cantidad de pruebas

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el sistema es simple y tiene una calidad aceptable, de manera que los atributos analizados se comportan de la manera siguiente:

Acoplamiento: según los resultados que se muestran, solo el 33% de las clases posee valores medios de acoplamiento, el resto son valores bajos, validando una realización correcta del diseño, al no existir un alto valor de dependencia entre clases.

Complejidad de mantenimiento: según los resultados que se muestran en la figura anterior, el 100% de las clases se comportan de forma satisfactoria pues, son de fácil soporte.

Reutilización: según los resultados que se mostraron en la figura, el 88% de las clases tiene un alto grado de reutilización.

Cantidad de pruebas: luego de aplicar la métrica se obtuvo que el 88% de las clases poseen un bajo grado de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas de software.

Según lo analizado anteriormente, los valores de RC se comportan de forma satisfactoria. Estos resultados expresan que las clases del diseño de la herramienta presentan bajo acoplamiento, la complejidad de mantenimiento y la cantidad de pruebas son bajas y en consecuencia el grado de reutilización es alto.

3.4 Pruebas

En todo proceso de desarrollo de software es imprescindible la realización de pruebas para garantizar el buen funcionamiento y la calidad del producto final, así se demuestra el cumplimiento de los objetivos de la investigación. Entre las principales fortalezas de la metodología XP es el proceso de pruebas, al realizarse de manera efectiva y continua, garantizan que los errores sean detectados en un corto plazo de tiempo y se corrijan de forma sencilla, lo cual segura el éxito del producto final.

En la validación de la herramienta se utilizaron los niveles de pruebas de unidad y aceptación, definidos en el Capítulo 1. Para realizar las pruebas de unidad se aplicaron los métodos de caja negra y caja blanca. Las pruebas de aceptación fueron realizadas con el cliente.

3.4.1 Pruebas de unidad

Al desarrollar un nuevo software la primera etapa a considerar es la de las pruebas unitarias, también llamadas pruebas modulares ya que permiten determinar si un módulo del programa está listo y correctamente terminado, las mismas no se deben confundir con las pruebas informales que realiza el programador mientras está desarrollando **[33]**. Como parte de las pruebas unitarias se aplicaron a la herramienta los métodos de caja blanca y caja negra como se plantea anteriormente.

Pruebas de Caja Blanca:

El método de caja blanca garantiza que se ejerciten por lo menos una vez todos los caminos independientes del código, así como la ejecución de todos los bucles en sus límites operacionales y todas las decisiones lógicas en las vertientes verdaderas y falsas **[28]**. Para aplicar las mismas se empleó la técnica del camino mínimo. Se tomó como ejemplo el

método determinarProducciónTerminada, perteneciente a la clase ConfigProduccion.java como base para realizar el procedimiento anterior. La selección del método se realizó teniendo en cuenta la complejidad del mismo, el cual da lugar a una de las principales funcionalidades que responden al objetivo general de la investigación. Esta permitió obtener una medida de la complejidad lógica para el diseño de los casos de prueba (CP) y usar dicha medida como guía para la definición de un conjunto básico de caminos de ejecución.

A continuación, se describen los pasos que se realizaron para desarrollar la técnica del camino básico.

Confeccionar el grafo de flujo: usando el código de la figura 7 se realizó la representación del grafo de flujo, el cual está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos

En la Figura 17 se muestra el grafo de flujo obtenido:

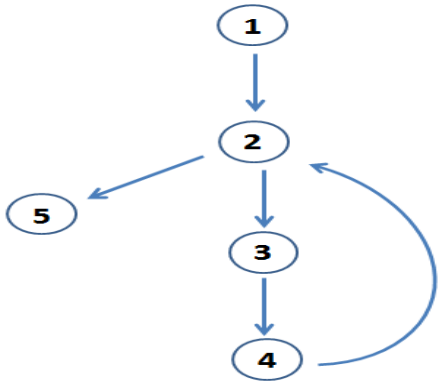

Figura 17: Prueba del camino básico

1. **Calcular la complejidad ciclomática:** proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, la complejidad ciclomática se calculó con la siguiente fórmula:

- V (G)= A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

V (G) = A-N+2 = 5-5+2 = 2

2. **Determinar un conjunto básico de caminos linealmente independientes:** el valor de V (G) da el número de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes 2 caminos:

Técnica del Camino Mínimo:

Camino básico 1: 1,2,3,4,2,5

Condición: Capacitar el personal

Resultado esperado: El sistema devuelve la producción terminada una vez capacitado el personal.

Camino básico 2: 1,2,5

Condición: Sin capacitar el personal

Resultado esperado: El sistema devuelve la producción terminada sin haber capacitado el personal.

3. **Obtención de casos de pruebas (CP):** cada camino independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. En este caso se obtuvieron 4 caminos básicos, por tanto, se hace necesario la confección de igual número de CP, para aplicar las pruebas a este método. A continuación, se muestran los casos de pruebas generados:

Caso de prueba: camino básico #1

Entrada	Capacitar el personal
Resultados esperados	El sistema devuelve la producción terminada una vez capacitado el personal.
Condiciones	Debe tenerse en cuenta el valor de APJ

Tabla 11: Caso de prueba camino básico #1

Caso de prueba: camino básico #2	
Entrada	No capacitar el personal
Resultados esperados	El sistema devuelve la producción terminada una vez se decida no capacitar el personal.
Condiciones	No se tiene en cuenta el valor de APJ

Tabla 12: Caso de prueba camino básico #2

Una vez ejecutados los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico, se concluye que los mismos fueron probados satisfactoriamente demostrando que el código generado no presenta ciclos infinitos y no existe código innecesario en el sistema desarrollado.

Pruebas de Caja Negra:

La técnica de diseño de prueba de caja negra es el procedimiento para obtener y/o seleccionar casos de prueba basados en el análisis de la especificación, tanto funcional como no funcional de un componente o sistema sin referencia a su estructura interna **[29]**. A continuación, se presenta el escenario que permite adicionar inversión en mantenimiento.

Descripción	Variable 1	Respuesta del sistema	Flujo central
Permite adicionar inversión en mantenimiento	*V(Introducir valores en los campos.)*	*El sistema muestra el mensaje "Acción realizada satisfactoriamente".*	*1. Seleccionar en el menú **Decisión*** *2. Seleccionar la opción **Mantenimiento*** *3. Introducir valores en el campo.* *4. Seleccionar la opción **Enviar**.* *5. El sistema agrega la inversión en mantenimiento a la base de datos.*
	I (introducir un string o carácter extraño en los campos.)	*El sistema muestra el mensaje "Existen valores incorrectos. Verifíquelos".*	
	I(Dejar campos vacíos)	*El sistema muestra el mensaje "Los campos no pueden estar vacíos".*	

Tabla 13: Caso de prueba del escenario adicionar inversión en mantenimiento

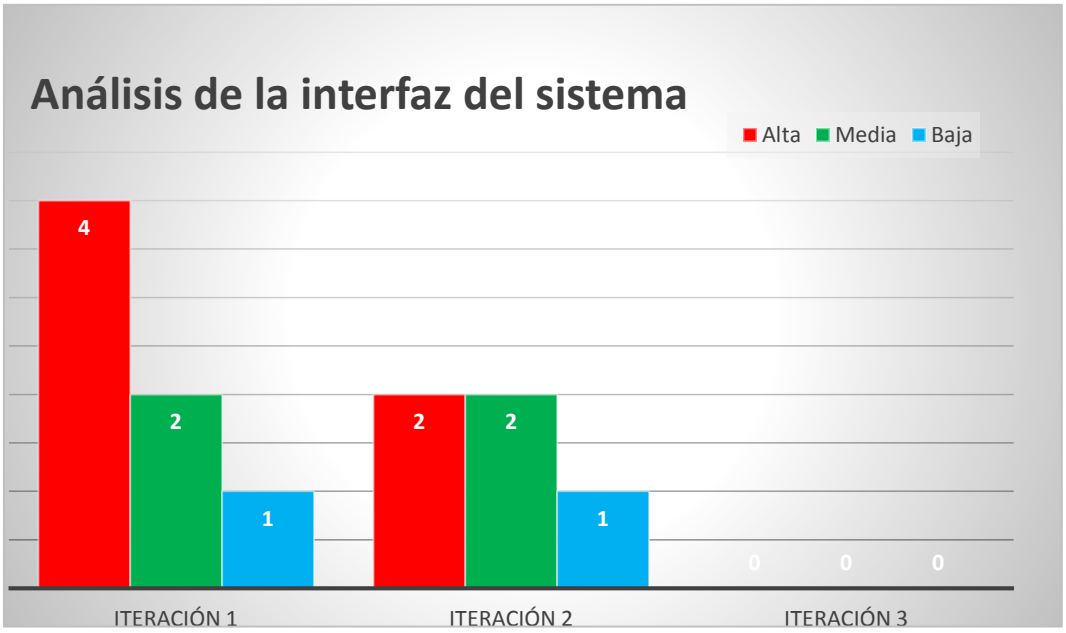

Figura 18: Análisis de las pruebas de caja negra

Con el objetivo de comprobar que las funcionalidades de la herramienta se realizaron correctamente y responden a las necesidades del cliente, el método se aplicó en tres iteraciones como se muestra en la Figura 18. En la primera se detectaron un total de 7 No Conformidades (NC), predominando entre ellas las de tipo interfaz, al finalizar la iteración todas estas NC quedaron resueltas. En la segunda iteración los resultados mejoraron al disminuir a 2 NC de importancia alta, ya en la tercera iteración se logró resolver las mismas

obteniéndose cero NC. La imagen anterior ilustra los resultados de aplicar el método de caja negra, teniendo en cuenta los tipos de NC identificados (interfaz, funcionalidad, redacción).

3.4.2 Pruebas de aceptación

La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar cómo el sistema satisface sus criterios de aceptación, validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio. Está considerada como la fase final del proceso, para crear un producto confiable y apropiado para su uso **[18]**.

Para la aplicación de estas pruebas se confeccionó un caso de prueba de aceptación por cada HU. Seguidamente se muestra el caso de prueba de aceptación de la HU “Determinar Producción Terminada”.

<table>
<tr><td colspan="3">Caso de prueba de aceptación</td></tr>
<tr><td colspan="2">Código de caso de prueba: 4</td><td>Nombre historia de usuario: Determinar Producción Terminada</td></tr>
<tr><td colspan="3">Nombre de la persona que realiza el caso de prueba: Osiel Sánchez Martínez</td></tr>
<tr><td colspan="3">Descripción de la prueba: revisar a través de la herramienta el correcto funcionamiento del RF Determinar Producción Terminada</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Se deben haber calculado los indicadores relacionados con la producción terminada.</td></tr>
<tr><td colspan="2">Entrada/Pasos de ejecución</td><td rowspan="2">Resultados esperados:</td></tr>
<tr><td>Acción:</td><td>Entrada</td></tr>
<tr><td>Se selecciona la opción de Determinar</td><td>1. Valor de la Capacidad Real</td><td>El sistema debe mostrar el resultado de la producción terminada para el período.</td></tr>
</table>

Producción Terminada.	2. Valor de inversión en mantenimiento 3. Valor de inversión en capacitación 4. Cantidad de materiales a producir	
Evaluación de prueba: Satisfactorio		

Tabla 14: Caso de prueba de aceptación de la HU "Determinar Producción Terminada"

Se realizó un encuentro con la Ing. Dailien Moré Soto, con el objetivo de revisar las funcionalidades de la herramienta, teniendo en cuenta los Casos de Pruebas definidos. Los resultados que se obtuvieron fueron satisfactorios, avalados por la especialista entrevistada, definiendo la aplicación como un importante aporte para el desarrollo de habilidades en los estudiantes de la asignatura de Simulación de Negocio de la carrera de Ingeniería Industrial en la CUJAE.

3.5 Conclusiones parciales

- La aplicación de técnicas de validación de requisitos, métricas de validación del diseño y pruebas unitarias, certifican la viabilidad del software, avalado por el acta de liberación.
- La aplicación de pruebas de aceptación a la solución propuesta, certifican la obtención de un software funcional que responde a los requerimientos del cliente, avalado por el acta de aceptación

CONCLUSIONES

- El estudio de los conceptos asociados a la investigación permitió definir buenas prácticas, analizar las metodologías y tecnologías utilizadas en el desarrollo de software, para seleccionar las más adecuadas a las características de la herramienta propuesta.
- El estudio de los simuladores de negocios con fines docentes y las soluciones existentes en la actualidad destinadas al área de producción, constituyen la base para el desarrollo del modelo matemático de la presente investigación. Permitió identificar la función de interpolación polinomial como método para el cálculo de los indicadores AJL y CDT.
- El desarrollo del módulo de producción del simulador con fines docentes SINEG permite estimar la producción terminada aprovechando los datos asociados a los factores productivos en la asignatura de Simulación de Negocio de la CUJAE.
- La aplicación de técnicas de validación de requisitos, métricas de validación del diseño, pruebas unitarias y de aceptación a la solución propuesta, certifican la obtención de un software funcional que responde a los requerimientos del cliente, avalado en cada caso por las actas de liberación y aceptación emitidas.

RECOMENDACIONES

Durante la realización de la presente investigación surgieron ideas que pueden servir como recomendación para el perfeccionamiento del módulo:

- Realizar una planificación de los recursos materiales a utilizar durante el proceso productivo, para garantizar la existencia de los materiales y materia prima en las áreas asignadas a la línea de producción.

BIBLIOGRAFÍA REFERENCIADA

[1] O. L. W. A. Raunk MS, «Developing discrete event simulations frm rigorous process definitions,» *Proceedings of the 2011 Symposium on Theory of Modeling and Simulation.,* pp. 117-124, 2011.

[2] S. Castro, «Juegos, Simulaciones y Simulación- Juego y los entornos multimediales en educacion Mito o potencialidad,» *Revista de Investigacion Universiddad Pedagógica Experimental Libertador,* pp. 223-245, 2008.

[3] J. y. G. F. A. I. Alonso González, *Características fundamentales del software dedicados a la enseñanza. Curso Informática Educativa. CESoftE.,* La Habana, 1994.

[4] B. Mendez Rodríguez, «Simuladores de Negocios en apoyo al aprendizaje. Facultad de Contaduría y Administración.,» 2012. [En línea]. Available: http://cdigital.uv.mx/bitstream/123456789/31665/1/mendezrodriguezberenice.pdf.

[5] J. y. P. G. A. J. Gimeno Sacristán, Comprender y transformar la enseñanza, Madrid: Morata S. A, 1992.

[6] V. González Castro, Teoría y práctica de los medios de enseñanza, La Habana: Pueblo y Educación, 1990.

[7] R. Shannon, Simulación de sistemas: Diseño, desarrollo e implementación., Mexico: Trillas, 2010.

[8] E. G. Zavaleta, «El uso de los simuladores de negocios en el proceso de aprendizaje.,» *Departamento de Negocios y Administración,* 2001.

[9] C. Biometría, «MODELOS Y SIMULACION,» *Informática Aplicada, Cátedra de Calculo Estadísitico y Biometría,* 2010.

[10] A. Torres, «El simulador de negocios como medio de capacitacion al personal de una empresa,» Universidad Veracruzana, 2010.

[11] F. a. Guillem, «Estudio de mercado sobre simuladores empresariales,» 2014.

[12] A. Marín, «CompanyGame,» *Oferta formativa basada en Simuladores de negocio,* 2015.

[13] I. Morales García, Metodologías de desarrollo software. ¿Tradicional o Ágil?, 2015.

[14] A. J. D. F. M. a. J. M. Navarro Cadavid, Revisión de metodologías ágiles para el desarrollo de software, 2013.

[15] F. E. a. R. B. Valderrama Guayan, Desarrollo de un sistema informático web para la gestión de producción de calzados de la empresa Jaguar S.A.C. utilizando la metodología AUP y tecnología ASP.NET framework MVC3., 2014.

[16] E. M. M. A. Y. J. G. a. X. L. Gallo, «Informe valorativo acerca del uso de metodologías ágiles en comunidades de desarrollo de software libre.,» 2010.

[17] R. S. Pressman, Ingeniería del Software., España, 2010.

[18] M. O. P. M. Rodríguez Corbea, «LA METODOLOGÍA XP APLICABLE AL DESARROLLO DEL SOFTWARE EDUCATIVO EN CUBA,» La Habana, 2007.

[19] R. S. Pressman, Ingeniería del Software. Un Enfoque Práctico. Quinta Edición., España: McGraw-Hill, 2002.

[20] D. E. Kanagusico Hernández, Introducción a la programación., 2010.

[21] N. a. S. C. Diakopoulos, «"Interactive: The Top Programming Languages 2015.",» 2015. [En línea]. Available: http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2015..

[22] E. Castañeda Sanabria, Java Básico, 2012.

[23] A. Sánchez, "Entorno Desarrollo Integrado (IDE)", 2014.

[24] S. J. Vallejos, «Sistemas de BD en Dispositivos Móviles y su integración con las BD tradicionales,» Universidad Nacional del Nordeste, 2009.

[25] J. Doren, «Administradores de BD para SQLite,» 2015. [En línea]. Available: http://soyprogramador.liz.mx/category/data-base/sqlite/.

[26] E. R. H. R. J. a. J. V. Gamma, Design Patterns.Elements of Reusable Object-Oriented Software., 1994.

[27] R. S. Pressman, Ingeniería del Software. Un enfoque práctico. Sexta Edición, España: Mc Graw Hill, 2003.

[28] J. Zapata, Niveles de prueba del software., 2013.

[29] P. a. M. C. P. Letelier, « "Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).",» 2006. [En línea]. Available: http://www.cyta.com.ar/ta0502/v5n2a1.htm..

[30] A. M. Alvarez, «Qué es MVC.,» 4 2 20014. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html.

[31] Ortega Paredes, José Grabiel , «Contabilidad. Paradigma de reconstrucción a través del giro informático,» Madrid, EAE, 2012, pp. 33-37.

[32] «Base de Datos: Tercera Formal(3FN),» 29 8 2015. [En línea]. Available: http://tadebasegino.blogspot.com/2012/08/tercera-forma-normal-3fn.html.. [Último acceso: 16 4 2017].

[33] A. Oré, UNIT TESTING - PRUEBAS UNITARIAS - CAP 1, 2009.

[34] M. Dunna, SIMULACIÓN Y ANÁLISIS DE MODELOS ESTOCASTICOS, Mexico: MCGRAW-HILL, 1996.

[35] A. Derby, «Apache Derby,» 4 marzo 2016. [En línea]. Available: https://db.apache.org/derby/.

[36] J. I. y. G. C. M. A. Pozo, «Aprender y enseñar ciencia,» Madrid, Morata S.L, 2001.

[37] R. Canales Reyes, «Identificación de factores que contribuyen al desarrollo de actividades de enseñanza y aprendizaje con apoyo de las TIC, que resulten eficientes y eficaces,» Barcelona, 2006.

[38] L. a. P. P. Calabria, MetodologíaXP, 2003.

[39] J. M. Santibáñez, Fundamentos de las metodologías en la ingeniería del software., 2015.

[40] Zambrano, ¿Qué son las tarjetas CRC?, 2014.

[41] F. Ruiz, « "Software Architecture.",» 2011. [En línea]. Available: http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/concepts/software_architecture,__O7tAMVvEduLYZUGfgZrkQ.html.

1. BIBLIOGRAFÍA CONSULTADA

1. Rodríguez Illera, J. (1997). El aprendizaje mediado con ordenadores: realidades contextuales y Zona de Desarrollo Próximo. En Revista Cultura y Educación. N. 6–7, 1997. pp. 77–90. Madrid.

2. Romero Morante, J. (2000). Del diseño y producción de medios al uso pedagógico de los mismos en las nuevas tecnologías de la información en la educación, pp. 15–30. Pons, J. y Gortarí Drets, C. Ediciones Alfar, S. A. Sevilla.

3. Santángelo, H. N. (2000). Modelos pedagógicos en los sistemas de enseñanza no presenciales basados en nuevas tecnologías y redes de comunicación. En Revista iberoamericana de educación, N. 24, pp. 135–159. Madrid.

4. Sevillano García, Ma. L. (1995). Las nuevas tecnologías como medios instructivos. En El curriculum: fundamentación, diseño, desarrollo y evaluación. pp 259–294. Madrid.

5. Hurtado, G.P.G. (2006). "Solo requisitos 2006". Volumen, 37.

6. Trahtemberg, L. (2000). El impacto previsible de las nuevas tecnologías en la enseñanza y la organización escolar. En Revista iberoamericana de educación, N. 24, pp. 37–62. Madrid.

7. Valdés Castro, P. [et. al.]. (1999). El proceso de enseñanza - aprendizaje de la Física en las condiciones contemporáneas. Editorial Academia: Colección alsi. La Habana.

8. Anijovich, R; Cappelletti, G; Mora, S. y Sabelli, Mª J. (2009). Transitar la formación pedagógica. Dispositivos y estrategias. Buenos Aires: Paidós.

9. Thayer, M.D.a.R. (1990). "Standards, Guidelines and Examples on System and Software Requirements Engineering".

10. Carratala, J. (2003). Administración de la empresa con Microsoft Excel. Omicron System. Buenos Aires.

11. Sureda Negre, J. (1986). La simulación como acción tecnológica en educación ambiental. En Revista Teoría de la Educación. N. 1, Málaga.Adell, J. (1997).

12. Chan Núñez, Ma.E. (2004). Propuesta metodológica para el análisis de las

competencias mediacionales en procesos educativos en entornos digitales. Tesis Doctoral. Universidad de Guadalajara. Guadalajara. México.

13. Díaz Barriga, F. (2005). Principios de diseño instruccional de entornos de aprendizaje

apoyados con TIC: un marco de referencia sociocultural y situado. [versión electrónica]. Revista Tecnología y Comunicación Educativas. 41. Recuperado el 9 de abril de 2008, de: http://cursa.ihmc.us/rid=1197697109500_1928608710_8051/c56art1.pdf

14. Galeano, M.E. (2004). Diseños de proyectos en la investigación cualitativa. Medellón.

Fondo editorial: EAFIT.

15. Goetz, J.P y LeCompte, M.D. (1988). Etnografía y diseño cualitativo en investigación

educativa. España. Morata.

16. Hernández Sampieri, R. Fernández-Collado, C y Baptista Lucio, P. (2008).

Metodología de la Investigación. (4ta. Ed.). México: McGraw-Hill Interamericana.

17. Latorre, A; Del Rincón, D. Y Arnal, J. (2003). Bases metodológicas de la investigación

educativa. Barcelona, España: Ediciones Experiencia, S.L.

18. Crahay, M. (2002). Psicología de la educación. Editorial Andrés Bello. Santiago de Chile.

19. Mesa, R. (1995). La Formación Administrativa Apoyada en Juegos Gerenciales.

Revista Universidad EAFIT. Nº98, 43-57. Recuperado el 9 de abril de 2014, de: http://publicaciones.eafit.edu.co/index.php/revista-universidad-eafit/article/view/1245/1130::pdf

20. Navarro, Rubén y García Santillán, A. (2009). Un modelo didáctico basado en el diseño de simuladores: el caso de la matemática financiera. Revista Ide@s CONCYTEG. Año 4, Núm. 46, 14 de abril de 2009. pp. 662,663.