Universidad de las Ciencias Informáticas

Facultad 2

Herramienta de apoyo a la enseñanza de la Combinatoria y Teoría de Números en la Asignatura Matemática Discreta.

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

Autores: Lianet García Ricardo.
Jorge Manuel Osorio Peña.

Tutor: Ing. Angélica M. Díaz Valdivia.
Co-tutor: Ing. Edgar González Blanco.

La Habana 19 julio del 2013

… las cosas simples deben ser simples, las cosas complejas deben ser posibles...
Alan Kay

DECLARACIÓN DE AUTORÍA.

Declaro que soy el único autor de este trabajo y autorizo al centro de Informatización para la Seguridad Ciudadana de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

"[Insertar nombre(s) de autor(es)]"

_____

"[Insertar nombre(s) de tutor(es)]"

_____

DATOS DE CONTACTO.

Tutor: Ing. Angélica M. Díaz Valdivia. **Email**: angelica@uci.cu

Síntesis del Tutor:

Graduado en el 2006 de Ingeniería en Informática en la UPR. Experiencia de 7 años en la impartición de asignaturas de la disciplina Matemáticas en el departamento de Ciencias Básicas. Actualmente cursa estudios de doctorado en Matemáticas en la Universidad de La Habana.

AGRADECIMIENTOS.

Agradecimientos Osorio

Primeramente quisiera agradecer muy especialmente a Mario Iván Cid Vázquez(Mayito)

A mis amigos del Yoagner (Yoa), El Zurdo, Alejandro (Ale), Ivo Daniel, Dandier, Luis Carlos, Dany, Lázaro (Lachy), por cuidar de mí y apoyarme en todo momento, en las buenas y en las malas.

A mis compañeros de escuela Yordanis, Karel,Dausbel, Yadira, Prada, Elías, Gual y otros que ya no se encuentran aquí como Alfredo (El Firi), José Miguel, Ana Karla, por brindarme su amistad desinteresadamente y ayudarme en los momentos difíciles de mi carrera.

A mis amigos del gimnasio Alexander, Hurshel, Medinilla, Guillermo, Alison.

A mis amigos de las BUTS Reinier (El Negro), Humberto, Rozney, Josue.

A los profes Alexeis, Gilberto, Lex Karel, Jorgito, Alejandro.

A mis tutores Angélica y Edgard.

A Ronny, Niorges, Dagoberto y Daniela.

Agradecimientos Lili

Mi primer agradecimiento es para mi compañero de tesis Jorge Manuel Osorio, gracias Oso; eres el motivo por el que ahora escribo estas palabras.

A Biasmey por ser tan especial conmigo y quererme de la manera que lo haces. Eres lo mejor que me ha sucedido en la vida. Te Adoro.

A toda mi familia que es tan grande que no cabe en dos líneas. En especial a mi tía Lupe por creer en mí y alentarme en todo el camino hasta aquí.

Agradecimientos

A Prada y Yudelis por estar siempre preocupadas y atentas conmigo, por quererme tal y como soy y no abandonarme aun cuando sé que pensarlo hacerlo; las quiero mucho.

A Zoilita, Anelis y Dunnia mis mejores amigas de todos los tiempos, no las olvido.

A mis tutores Angélica y Edgar por todo su apoyo y por confiar en mí y darme la maravillosa oportunidad de convertirme en ingeniera.

A mis amigas de primer año Yordialis y Yuneimi, las extraño

Dedicatoria.

Osorio

Dedico mi tesis muy especialmente a mi familia y amigos, pero especialmente a cuatro pilares en mi vida ellos son:

Mi madre Odalis, por ser lo más maravilloso que tengo en esta vida, no sabría cómo devolverte todo el apoyo y el amor que me has dado, creo que no hay palabra en el mundo que describa todo lo que has hecho por mí.

Mi padre Jorge, aunque no estés aquí, quiero que sepas que te quiero muchísimo, cumplo tu sueño y tu deseo, donde quieras que estés cuídame siempre.

A mis primos Yeiler y Gabriel, más que primos somos hermanos, el primero que aunque no estés físicamente a mi lado, nunca olvidare los buenos momentos que pase contigo, y el segundo que más podría yo pedir de ti, si lo tengo todo.

Dedico también a mis primos Wilson, Denielita, Rolandito, Barbarita, Alejendrito, Yilian y Eliza.

A mis tías queridas Elaine, Ledais, Zarita, Clara, un besote para ti, Bárbara, Isabel.

A mis tíos Carlos y Manolin.

A mis abuelitos Belkis y Manuel.

A mi familia en general por el apoyo incondicional que me ha dado, a todos un beso.

Lili

Dedico mi tesis a mamá Tania, a mis papás Luis y Guillermo, son lo más especial que tengo.

A mi tía Iris, por estar siempre pendiente de mí.

A mi bisa linda.

A mi Abuela Miriam y a mi abuela Hilda.

A mi hermanito Luisito y mi hermana Olguita.

A mis tías y mis tíos.

A todo el que de una u otra manera contribuyeron en mi carrera; a todos los quiero mucho.

RESUMEN.

Los avances que han tenido lugar en estos últimos tiempos en el campo de las Matemáticas, con respecto a la resolución de problemas de la vida real, en gran medida se deben al vertiginoso desarrollo que ha experimentado la Informática y su interrelación con las Matemáticas, lo que ha permitido su perfeccionamiento y desarrollo, razón por la que ambas son vistas como ciencias complementarias. La Informática, viabiliza las herramientas con la cuales la Matemática se apoya para la modelación de fenómenos del mundo circundante, cada vez más complejos.

Es por ello que la constante inclusión de las Matemática así como el trabajo con asistentes matemáticos, se justifica aún más; pues permite el perfeccionamiento y fortalecimiento del proceso de enseñanza-aprendizaje, traducido como la formación del futuro profesional capaz de identificar e interpretar con pensamiento lógico, los hechos que acontezcan a su alrededor.

A nivel mundial, existen software que son usados para áreas específicas de las Matemáticas, por ejemplo, WinQSB en la Investigación de Operaciones, SPSS (Statistical Package For Social Sciences) y Statistic en la Estadística, Biometría, Econometría, entre otras. En el contexto cubano, no se cuenta con un software que asista al trabajo de los estudiantes que reciben como asignatura Matemática Discreta. Es por ello, que se pretende la elaboración de una herramienta, en la Universidad de las Ciencias Informáticas (UCI), que permita el trabajo con objetos discretos en los temas Combinatoria y Teoría de Números y está diseñado de manera didáctica para que favorezca el desempeño exitoso del estudiante en el proceso de enseñanza – aprendizaje de la asignatura.

PALABRAS CLAVE

Matemáticas, informática, asistente matemáticos, software.

ÍNDICE.

AGRADECIMIENTOS.....II

RESUMEN.....V

INTRODUCCIÓN.....1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.....5

1.1 Los asistentes matemáticos en el proceso de enseñanza-aprendizaje. 5
1.2 Asistentes matemáticos de mayor uso en el mundo..... 5
1.2.1 Derive 6..... 6
1.2.2 Maple 13 6
1.2.3 MatLab 6
1.2.4 Mathematica..... 7
1.3 Herramientas existentes para el trabajo con la Combinatoria y Teoría de Números en la UCI 7
1.3.1 Asistente Matemático: MathematicaVersión: 8.0 8
1.3.2 Asistente Matemático: DeriveVersión: 6.00..... 9
1.4 Matemática Discreta. Teoría de la Combinatoria y Teoría de Números. 12
1.4.1 Teoría de la Combinatoria. Principales conceptos y definiciones 13
1.4.2 Teoría de Números. Principales conceptos y definiciones..... 17
1.5 Metodología de desarrollo. 20
1.5.1 Metodología tradicional. Proceso Unificado de Desarrollo (RUP) 20
1.5.2 Metodologías ágiles.Extreme Programming (XP) 22
1.6 Herramientas y Tecnologías. 23
1.6.1 Lenguaje de programación. Java 23
1.6.2 Entorno de Desarrollo Integrado. NetBeans..... 24
1.6.3 Herramienta Case. Visual Paradigm 25
1.6.4 Lenguaje de Modelado. UML 26
1.6.5 Conclusiones parciales del capítulo 27

CAPÍTULO 2: PLANIFICACIÓN Y DISEÑO DE LA HERRAMIENTA. 28

Introducción..... 28
2.1 Fase Planificación..... 28

2.1.1 Propuesta de solución. 29
2.1.2 Especificación de los requisitos de software 31
2.1.3 Historias de Usuarios (HU) 33
2.1.4 Estimación del Esfuerzo por Historia de Usuario 37
2.1.5 Plan de Iteraciones 39
2.1.6 Plan de Entrega 42
2.2 Fase de Diseño 44
2.2.1 Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración) 45
2.3 Conclusiones parciales del capítulo 48
CAPÍTULO 3: DESARROLLO Y PRUEBAS DE LA HERRAMIENTA 49
Introducción 49
3.1 Fase de Desarrollo 49
3.1.1 Arquitectura del Sistema 49
3.1.2 Patrones de Diseño 51
3.1.3 Tareas 55
3.2 Fase de Pruebas 59
3.3 Conclusiones parciales del capítulo 62
CONCLUSIONES GENERALES. 63
RECOMENDACIONES. 64
BIBLIOGRAFÍA. 65

Índice de Figuras y Tablas

Índice de Figuras y Tablas.

Tabla 1. Descripción de la HU Determinar la cantidad de permutaciones de n elementos 35
Tabla 2. Descripción de la HU Generar todas las permutaciones de un conjunto dado 35
Tabla 3. Descripción de la HU Determinar la cantidad de combinaciones posibles de r elementos de n 36
Tabla 4. Descripción de la HU Generar todas las combinaciones de un conjunto dado 36
Tabla 5. Descripción de la HU Determinar la cantidad de permutaciones posibles de r elementos de n 37
Tabla 9. Puntos de Estimación 38
Tabla 10. Plan de duración de las Iteraciones 41
Tabla 11. Plan de Entregas 42
Tabla 12. Clase: CalculadorValores 45
Tabla 13. Clase: CalculadorConjuntos 46
Tabla 14. Clase: CalculadorEnteros 46
Tabla 15. Clase: ConversorNúmeros 46
Tabla 16. Tareas de la Iteración 1 56
Tabla 17. Tareas de la Iteración 2 56
Tabla 18. Tareas de la Iteración 3 57
Tabla 19. Tareas de la Iteración 4 58
Tabla 20. Tarea 1: Crear funcionalidad Determinar cantidad de permutaciones de n elementos 59
Tabla 28. Caso de Prueba de Aceptación HU1_P1 60
Tabla 29. Caso de Prueba de Aceptación HU2 61
Tabla 30. Caso de Prueba de Aceptación HU4 62

Figura 1. Proceso de desarrollo de software 21
Figura 2. Representación de las fases de la Metodología XP 23
Figura 3. Integración de Visual Paradigm en el proceso de desarrollo. 26
Figura 4. Programación Extrema 28
Figura 5. Propuesta de solución 30
Figura 6. Prototipo de Interfaz 30
Figura 7. Diagrama de Clases del Sistema 47
Figura 8. Arquitectura N-Capas 50
Figura 9. Arquitectura de la herramienta 51
Figura 10. Ejemplo patrón SingletonenlaClaseCalculadorConjuntos 53
Figura 11. Patrón Singleton 53
Figura 12. Ejemplo patrón Controlador en la Clase ControladorSistema 54

INTRODUCCIÓN.

En la actualidad los sistemas educativos de todo el mundo se enfrentan al desafío de utilizar las Tecnologías de la Información y las Comunicaciones para proveer a los alumnos con las herramientas y conocimientos necesarios que se requieren en el siglo XXI. El diseño e implementación de programas de capacitación docente que utilicen las Tecnologías de la Información y las Comunicaciones (TIC's) constituye un elemento clave para lograr reformas educativas profundas y de amplio alcance. Las instituciones de formación docente deberán optar entre asumir un papel de liderazgo en la transformación de la educación, o bien quedar atrás en el continuo cambio tecnológico. Para que en la educación se puedan explotar los beneficios de las TIC's en el proceso de aprendizaje, es esencial que tanto los futuros docentes como los docentes en actividad sepan utilizar estas herramientas. (1)

La incorporación de las TIC's en la sociedad y en especial en el ámbito de la educación ha ido adquiriendo una creciente importancia, evolucionando lo largo de estos últimos años, tanto que la utilización de estas tecnologías en el aula pasa de ser una posibilidad a erigirse como una necesidad y como una herramienta de trabajo básica para el profesorado y el alumnado. (2)

Las Tecnologías de la Información y las Comunicaciones (TIC's) que como resultado del desarrollo tecnológico y de las nuevas transformaciones del sistema educacional cubano se introducen en Cuba, contribuyen al perfeccionamiento del proceso de enseñanza-aprendizaje. En las universidades constituye una herramienta de apoyo al proceso docente-educativo, siendo la Universidad de las Ciencias Informáticas una de las instituciones que utiliza este tipo de aplicaciones ya que su sistema de enseñanza está basado en el uso de las TIC's. (1)

La relación computación-matemática contiene varias corrientes. Una de ellas es la vertiente o tendencia de la aplicación de las matemáticas (principalmente el álgebra booleana, las matemáticas discretas, la teoría de grafos, la lógica matemática, la probabilidad y estadísticas) a la computación. Como complemento indispensable de esta relación, existen asistentes matemáticos, siendo estos de vital importancia para fomentar el vínculo entre estas dos ciencias. (2)

El trabajo surge con la necesidad de elaborar una herramienta informática con características de un asistente matemático que apoye a la asignatura de Matemática Discreta que es la parte de la matemática encargada del estudio de los conjuntos discretos: finitos o infinitos numerables. Estudia estructuras cuyos

elementos pueden contarse uno por uno separadamente, sin dar lugar a números decimales ni procesos infinitos. Es decir, los procesos en matemática discreta son finitos y contables. (3)

Actualmente no existe en la UCI una herramienta que solo permita el trabajo con objetos discretos, específicamente en los temas de Combinatoria y Teoría de Números. Las herramientas que se utilizan en la universidad, no desarrollan completamente las funcionalidades requeridas en los temas Combinatoria y Teoría de Números especificadas en el programa analítico de la asignatura.

Por todo lo antes expuesto el **problema a resolver** queda resumido en la siguiente interrogante: ¿Cómo desarrollar una herramienta informática que supla las carencias presentes en los asistentes matemáticos utilizados en la UCI, para la enseñanza de la Combinatoria y Teoría de Números en la asignatura Matemática Discreta?

El **objeto de estudio** se encuentra enmarcado en: La Teoría de Combinatoria y la Teoría de Números en la asignatura Matemática Discreta.

El **campo de acción** se encuentra definido en: Los asistentes matemáticos existentes en el proceso de enseñanza-aprendizaje de la Matemática Discreta.

Se define como **objetivo general**: Desarrollar una herramienta informática, que asista al trabajo con objetos discretos, con un enfoque de apoyo en la enseñanza de la Combinatoria y la Teoría de Números en la asignatura Matemática Discreta en la Universidad de las Ciencias Informáticas.

Preguntas de investigación:

- ✓ ¿Cuáles son las principales funcionalidades que no implementan las herramientas existentes para el trabajo con la Combinatoria y Teoría de Números, en las Matemáticas Discretas?
- ✓ ¿Cómo definir, diseñar e implementar los módulos del asistente matemático (y sub-módulos, en caso necesario) y sus funcionalidades, a partir de los temas de la asignatura Matemática Discreta?
- ✓ ¿Cómo elaborar una guía didáctica para el uso del asistente matemático en la asignatura Matemática Discreta?

Para dar solución al problema planteado se desglosan los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación para lograr la correcta selección de la metodología y las tecnologías a utilizar en el desarrollo de la solución.
- ✓ Realizar un levantamiento de las funcionalidades de un grupo de asistentes matemáticos, sobre el tratamiento de objetos discretos, en los temas Combinatoria y Teoría de Números.
- ✓ Conformar los requisitos funcionales y no funcionales de la herramienta de apoyo a la enseñanza de la Combinatoria en la asignatura Matemática Discreta.
- ✓ Especificar las herramientas, lenguaje de programación y metodología de desarrollo de software a utilizar para la elaboración de una herramienta que apoye la enseñanza del contenido Combinatoria y Teoría de Números a los estudiantes de la UCI.
- ✓ Modelar el análisis y diseño de la herramienta.
- ✓ Implementar el sistema con las características definidas en los procesos de análisis y diseño.
- ✓ Realizar pruebas que validen la herramienta desarrollada.

Para cumplir con los objetivos específicos se proponen como **tareas de investigación**:

- ✓ Análisis de las principales herramientas, metodologías y tecnologías para la solución del problema.
- ✓ Análisis del estado del arte sobre los asistentes matemáticos más usados en la enseñanza, a nivel internacional, así como los de mayor uso en la Universidad de Ciencias Informáticas.
- ✓ Analizar cómo efectúan el tratamiento de objetos discretos los asistentes matemáticos que se emplean en la Universidad de Ciencias Informáticas.
- ✓ Análisis de la arquitectura definida por el equipo de desarrollo de la Herramienta de apoyo a la enseñanza de la Combinatoria y Teoría de Números en la Asignatura Matemática Discreta.
- ✓ Implementación de los módulos correspondientes de acuerdo a las normas y estándares establecidos.
- ✓ Diseño y aplicación de pruebas a los módulos implementados para validar la solución propuesta.

Los métodos teóricos y empíricos utilizados a lo largo de la investigación son los siguientes:

Métodos Teóricos:

- ✓ **Analítico – sintético**: Se examinarán bibliografías correspondientes a los contenidos de Matemática Discreta I y Matemática Discreta II, con el de objetivo de utilizar la información para la realización de una herramienta para el apoyo en la enseñanza de la Combinatoria y la Teoría de Números.
- ✓ **Modelación**: Se utilizará para confeccionar modelos que representen de forma simplificada la aplicación a desarrollar, utilizar la abstracción para explicar la realidad que se quiere representar.

Métodos Empíricos:

- ✓ **Observación**: Se utiliza para constatar las deficiencias existentes en asistentes matemáticos utilizados actualmente en la UCI, así como adquirir conocimientos sobre el funcionamiento, ventajas y desventajas de otras aplicaciones educativas similares a la que se desarrollará.
- ✓ **Entrevista**: Es de beneficio para acumular opiniones de los profesores de Matemática Discreta en cuanto a las deficiencias observadas en los asistentes matemáticos utilizados en la UCI, además para recoger otras informaciones importantes referentes a las tecnologías usadas en la asignatura Matemática Discreta, con el fin de lograr la vinculación del producto final con la asignatura.

El documento está estructurado en tres capítulos que incluyen los siguientes elementos:

- ✓ **CAPÍTULO 1. Fundamentación Teórica:** Muestra un estudio detallado de la situación actual de los asistentes matemáticos en el contexto nacional e internacional, así como conceptos esenciales para un mejor entendimiento de la problemática.
- ✓ **CAPÍTULO 2. Planificación y Diseño:** Se describirán las características y se realizará una propuesta de la herramienta a desarrollar, se realiza una descripción detallada de la propuesta de solución, las especificaciones de requisitos de software, se describen las Historias de Usuario, se realiza el Plan de Iteraciones y de Entrega, los Requerimientos Funcionales y No Funcionales.
- ✓ **CAPÍTULO 3.Implementación y Pruebas:** Abordará las actividades definidas para la implementación y pruebas de la herramienta. Se expone el marco de aplicación, así como la arquitectura que tendrá la misma, implementación y pruebas a la herramienta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

Introducción del Capítulo.

En este capítulo se realiza la fundamentación teórica del tema a desarrollar a través de un estudio del estado del arte, donde se describe la influencia de los asistentes matemáticos en el proceso de enseñanza-aprendizaje haciendo énfasis en las aplicaciones, funciones y características que presentan los mismos. Se hace una breve exposición de algunos conceptos relacionados con la Teoría de Combinatoria y Teoría de Números para un mejor entendimiento de la propuesta de solución.

1.1 Los asistentes matemáticos en el proceso de enseñanza-aprendizaje.

Los asistentes matemáticos son recursos tecnológicos que permiten al estudiante concentrar esfuerzos en el razonar, solucionar y formular problemas, así como en verificar teoremas y propiedades matemáticas. Fortalecen el estudio, búsqueda, indagación, sistematización, socialización y experimentación de trabajos de aula, tendientes a mejorar la enseñanza y el aprendizaje de las matemáticas. Además mantienen una actitud positiva hacia las actividades docentes, al mismo tiempo que sean capaces de desarrollar en los contextos educativos como procesos de innovación, valorando la incidencia real de las tecnologías en la práctica docente cotidiana integrando rápidamente en todos los ámbitos laborales. (4)

Se puede considerar al asistente matemático como facilitador en el proceso de enseñanza-aprendizaje, es decir, posee una alta cualidad que los distingue de otros medios tradicionales. Esta cualidad permite recibir información, procesar y brindar una respuesta efectiva que contribuye en la interpretación de los fenómenos numéricos investigados, por esta razón se considera un medio poderoso en la enseñanza aprendizaje de la matemática que estimula a experimentar nuevas formas de aprender a pensar en lo investigativo, aprovechando el conjunto de potencialidades de todos los recursos asignados en la Educación Superior, pues la combinación de medios produce un resultado superior en la aplicación aislada de los mismos.(4)

1.2 Asistentes matemáticos de mayor uso en el mundo

Para el desarrollo de este trabajo se evaluaron diferentes asistentes matemáticos con el objetivo de conformar una idea general de estas aplicaciones. A continuación se presentan algunos de ellos:

1.2.1 Derive 6

Derive 6 es un poderoso sistema para hacer operaciones matemáticas simbólicas y numéricas. Procesa variables algebraicas, expresiones, ecuaciones, funciones, vectores, matrices y expresiones booleanas como una calculadora de procesos científicos. Los problemas en los campos de la aritmética, álgebra, trigonometría, cálculo, álgebra lineal y cálculo proposicional se pueden resolver con un clic del ratón. Grafica expresiones matemáticas en dos y tres dimensiones utilizando diferentes sistemas de coordenadas. Por su perfecta integración de las capacidades numéricas, algebraicas y gráficas, Derive es una excelente herramienta y uno de los mejores asistentes matemáticos. (5)

1.2.2 Maple 13

Maple es la herramienta de software líder en todas las materias matemáticas. Maple ofrece un motor de computación matemática avanzado y de alto desempeño, junto con varias herramientas numéricas y simbólicas completamente integradas.

Características:

- ✓ Completo soporte de documentación de gráficos, el cual le permitirá agregar información adicional a sus gráficos 3D, haciéndolos más atractivos y mucho más fáciles de interpretar. Las anotaciones disponibles incluyen títulos y etiquetas que contienen texto, expresiones matemáticas, flechas, dibujos a mano alzada, figuras geométricas, etiquetas de múltiplos de pi (π) y ejes coloreados individualmente.
- ✓ Al utilizar controles de rotación completa, puede cambiar interactivamente la visualización utilizando los tres ángulos de rotación, lo que le permite enfocar los puntos de interés del gráfico con facilidad.
- ✓ Puede crear gráficos 3D y 2D que involucren unidades. Estas unidades serán agregadas automáticamente como información adicional a los ejes de sus gráficos.(6)

1.2.3 MatLab

MATLAB es el nombre abreviado de “MATrixLABoratory”. MATLAB es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con números escalares −tanto reales como complejos−, con cadenas de caracteres y con otras estructuras de información más complejas. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en

dos y tres dimensiones. MATLAB tiene también un lenguaje de programación propio. MATLAB es un gran programa de cálculo técnico y científico. (7)

MATLAB es un lenguaje de alto rendimiento para la informática técnica. Integra cómputo, visualización y programación en un formato fácil de usar, en un entorno donde los problemas y soluciones se expresan familiarizados en notación matemática. Los usos típicos incluyen:

- ✓ Matemáticas y computación.
- ✓ Algoritmo de desarrollo.
- ✓ Modelado, simulación y prototipado.
- ✓ Análisis de datos, exploración y visualización.
- ✓ Ciencia y la Ingeniería Gráfica.
- ✓ Desarrollo de aplicaciones, incluyendo la construcción de interfaces gráficas de usuario.(7)

1.2.4 Mathematica

Casi cualquier flujo de trabajo incluye resultados de computación, y eso es lo que hace Mathematica desde la construcción de un sitio web de comercio de fondos de cobertura o la publicación de libros de texto de ingeniería interactivos, desarrollo de algoritmos de reconocimiento de imágenes incrustadas o cálculo enseñanza. Es conocida como la aplicación final del mundo para sus cálculos. Pero es mucho más: es la única plataforma de desarrollo de la plena integración de la computación en flujos de trabajo completos, que se mueve sin problemas desde las ideas iniciales hasta el final al individuo desplegado o soluciones empresariales. (8)

1.3 Herramientas existentes para el trabajo con la Combinatoria y Teoría de Números en la UCI

De los asistentes utilizados en la Facultad 2 de la UCI para el trabajo con objetos discretos, se escogieron dos: Mathematica y Derive siendo este último el de mayor uso. Obteniéndose de estos un listado de funcionalidades en las que queda reflejado que dichos asistentes no implementan en su totalidad los contenidos de Combinatoria y Teoría de Números, que se imparten en la asignatura de Matemática Discreta .A continuación se muestran funcionalidades de los asistentes escogidos, donde se marca con una cruz en la primera columna si el asistente matemático que se analiza realiza tal funcionalidad, además de reflejar observaciones sobre los temas en cuestión y finalmente arroja un resultado ,expresando en por ciento el cumplimiento de las mismas.

1.3.1 Asistente Matemático: Mathematica Versión: 8.0

Combinatoria.

AF	Funcionalidad
x	1. Determinar la *cantidad de permutaciones* posibles de n elementos: P(n).
x	2. Determinar todas las *permutaciones* de un conjunto dado.
x	3. Determinar la *cantidad de permutaciones* posibles de r elementos de n: P(n, r).
	4. Determinar todas las *permutaciones* de un subconjunto dado.
	5. Determinar la *cantidad de combinaciones* posibles de r elementos de n: C(n, r).
	6. Determinar todas las *combinaciones* de un conjunto dado.
x	7. Determinar la *cantidad de permutaciones* posibles de n elementos, con *repetición*.
x	8. Determinar todas las *permutaciones* de un conjunto dado, con elementos repetidos.
	9. Determinar la *cantidad de permutaciones* posibles de r elementos de n, con *repetición*.
x	10. Determinar todas las *permutaciones* de un conjunto dado, permitiendo *repetición* de elementos.
	11. Determinar la *cantidad de combinaciones* posibles de r elementos, con *repetición*, de n: C(n+r-1, r).
	12. Determinar todas las *combinaciones* con *repetición* de un conjunto dado.
	13. Trabajo con coeficientes binomiales.
	14. Representación del triángulo de Pascal.

Observaciones sobre combinatoria:

- ✓ Para calcular la cantidad de permutaciones de r objetos tomados de n se usa la función Length[Permutations[Range[n],{r}]] o bien se pone la formula: n!/(n-r)!
- ✓ Para determinar la cantidad de permutaciones de n objeto, se calcula Lenght [Permutations [Range [3]]] o bien se pone la fórmula: n!

- ✓ Para determinar la cantidad de permutaciones de un conjunto dado: Permutations [{Conjunto}]. Ejemplo: Permutations [{a, b, c, d}].
- ✓ Determinar todas las permutaciones de un conjunto dado, con elementos repetidos.: Permutations [{a,a,b}].

Porcentaje de cumplimiento: 42.86%.

1.3.2 Asistente Matemático: Derive Versión: 6.00

Combinatoria

AF	Funcionalidad
x	1. Determinar la *cantidad de permutaciones* posibles de n elementos: P(n).
	2. Determinar todas las *permutaciones* de un conjunto dado.
x	3. Determinar la *cantidad de permutaciones* posibles de r elementos de n: P(n, r).
	4. Determinar todas las *permutaciones* de un subconjunto dado.
x	5. Determinar la *cantidad de combinaciones* posibles de r elementos de n: C(n, r).
	6. Determinar todas las *combinaciones* de un conjunto dado.
	7. Determinar la *cantidad de permutaciones* posibles de n elementos, con *repetición*.
	8. Determinar todas las *permutaciones* de un conjunto dado, con elementos repetidos.
	9. Determinar la *cantidad de permutaciones* posibles de r elementos de n, con *repetición*.
	10. Determinar todas las *permutaciones* de un conjunto dado, permitiendo *repetición* de elementos.
	11. Determinar la *cantidad de combinaciones* posibles de r elementos, con *repetición*, de n: C(n+r-1, r).
	12. Determinar todas las *combinaciones* con *repetición* de un conjunto dado.
	13. Trabajo con coeficientes binomiales.
x	14. Representación del triángulo de Pascal.

Observaciones sobre combinatoria:

- ✓ Para calcular la cantidad de permutaciones de n objetos tomados de m se usa la función PERM (m, n).
- ✓ Para determinar la cantidad de permutaciones de n objetos, se hace m = n; de esta forma se calcula P(n) = n!
- ✓ El número de combinaciones de n objetos tomados de m se calcula mediante la función COMB (m, n).
- ✓ Aunque no existen funciones para determinar la cantidad de permutaciones y combinaciones con repetición, sí está definido el operador factorial, que permitiría construir una función para dichos cálculos.
- ✓ Utilizando la función COMB (m, n) se puede obtener, en forma de matriz, una cantidad específica de filas del triángulo de Pascal.

Porcentaje de cumplimiento: 28.6%.

Teoría de números

AF	Funcionalidad
x	1. Representar números en el *sistema binario*.
x	2. Representar números en el *sistema octal*.
x	3. Representar números en el *sistema decimal*.
x	4. Representar números en el *sistema hexadecimal*.
x	5. Realizar *conversiones* entre todos los sistemas de numeración.
x	6. Realizar *operaciones aritméticas* en todos los sistemas de numeración.
x	7. Permitir aritmética de trabajo con sistemas numéricos de *base variable*.
x	8. Determinar el *máximo común divisor* de dos enteros (mcd).
x	9. Determinar el *máximo común divisor* representado como una combinación lineal de dos enteros.
	10. Determinar si dos enteros son *primos relativos*.
x	11. Determinar el *mínimo común múltiplo* de dos enteros (mcm).
x	12. Determinar si un número entero dado es un *número primo*.
x	13. Descomponer en factores primos un número entero dado.
	14. Determinar si dos enteros son *congruentes*, dado un módulo.

x	15. Resolver ecuaciones lineales congruentes (*congruencias lineales*).
x	16. Resolver sistemas de congruencias lineales.
	17. Resolver ecuaciones diofánticas.
	18. Resolver sistemas de ecuaciones diofánticas.

Observaciones sobre teoría de números:

- ✓ Para cambiar la base de numeración en que se trabaja se usa la opción de menú: **Opciones**>**Ajustes de Modo**>**Introducción**. El campo **Base de Numeración** incluye los sistemas: binario, octal, decimal y hexadecimal. Por defecto se usa el sistema decimal.
- ✓ Se pueden definir bases usando cualquier entero entre 2 y 36, ambos inclusive.
- ✓ Para convertir números entre dos bases distintas o trabajar con números de base diferente a diez, se usan las expresiones **InputBase** := ***base*** y **OutputBase** := ***base***, donde ***base*** es Binary, Octal, Decimal, Hexadecimal o un entero entre 2 y 36, inclusive.
- ✓ Para determinar el máximo común divisor (mcd) de dos o más enteros se usa la función ***GCD*** (m_1, m_2,…, m_n). Esta función se simplifica al mcd de los números m_1, m_2,…, m_n.
- ✓ Aunque no es posible determinar cuándo dos números dados son primos relativos; sí se puede programar una función usando la del máximo común divisor.
- ✓ La función ***LCM*** (m_1, m_2,…, m_n) se simplifica al mínimo común múltiplo de los números m_1, m_2,…, m_n.
- ✓ Las funciones ***GCD*** y ***LCM*** pueden usar cualquier tipo de argumentos: números, vectores o matrices.
- ✓ Para determinar si un número dado es primo se usa la función ***PRIME?***(*m*), la que se simplifica a **true** si *m* es primo y a **false** si *m* no es primo; en otro caso no se simplifica.
- ✓ Otras funciones para el trabajo con los números primos son: ***NEXT_PRIME***(*m*), que se simplifica al siguiente número primo mayor que *m*, y ***PREVIOUS_PRIME***(*m*), que se simplifica al primer número primo menor que *m* si existe.
- ✓ Se tienen dos opciones para descomponer un número entero dado en sus factores primos: usar la opción de menú **Simplificar**>**Factorizar**, en el tipo de factorización escoger **Factores Primos**, o la función ***FACTOR***(*n*, Number).

- ✓ La función ***EXTENDED_GCD***(*a*, *b*) se simplifica al vector de enteros [g, [x, y]]donde g = ***gcd***(*a*, *b*) = x*a*+y*b*, o sea, permite obtener el máximo común divisor representado como una combinación lineal de los enteros *a* y *b*.
- ✓ Para resolver una ecuación lineal congruente se utiliza la función ***SOLVE_MOD***(*u*, *x*, *m*), que se simplifica al vector de soluciones de la ecuación ***u*(*x*)=0 módulo *m***.
- ✓ No existe una forma directa de determinar si dos números dados son congruentes, pero usando la programación en Derive es posible construir una función para dicho objetivo.
- ✓ La función ***CRT***(*a*,*m*) permite resolver sistemas de congruencias lineales de la forma $\mathbf{x = a_i \bmod m_i}$.
- ✓ El archivo **NumberTheoryFunctions.dfw**, en la carpeta de Derive **Users\NumberTheory**, define funciones adicionales de teoría de números.

Porcentaje de cumplimiento: 77.7%.

1.4 Matemática Discreta. Teoría de la Combinatoria y Teoría de Números.

La **Matemática Discreta** surge como una disciplina que unifica diversas áreas tradicionales de las Matemáticas (combinatoria, probabilidad, geometría de polígonos, aritmética, grafos,...),como consecuencia de, entre otras cosas, su interés en la informática y las telecomunicaciones: la información se manipula y almacena en los ordenadores en forma discreta (caracteres formados por ceros y unos), se necesita contar objetos (unidades de memorias, unidades de tiempo), se precisa estudiar relaciones entre conjuntos finitos (búsquedas en bases de datos), es necesario analizar procesos que incluyan un número finito de pasos (algoritmos). (9)

En términos más precisos, las **Matemáticas Discretas** estudian las propiedades de conjuntos y sistemas que tienen un número finito de elementos, o bien un número infinito numerable de elementos que estén separados entre sí. Y aquí es donde se debe precisar lo que quiere decir “estar separados entre sí”, lo que se traduce en que, para dos elementos distintos cualesquiera, existe un entorno de uno que no contiene al otro. (10) Definir el concepto *discreto* sin entrar en demasiadas formalidades no es sencillo, se puede apelar a ciertos ejemplos matemáticos conocidos y contraponerlo al concepto de *continuo*. Lo discreto es lo finito o lo que, si no es finito, presenta el aspecto de los números naturales, objetos bien separados entre sí; lo continuo es lo no finito, lo infinitesimalmente próximo, como los números reales, y de ahí el concepto de límite y las ideas que de dicho concepto se derivan. (11)

1.4.1 Teoría de la Combinatoria. Principales conceptos y definiciones

Dentro de la **Matemática Discreta** se analiza la **Teoría Combinatoria** que estudia las agrupaciones que pueden ser formadas cuando se toman todos, o algunos, de los elementos de un conjunto finito. Los elementos del conjunto pueden ser de cualquier naturaleza: números, personas, empresas, artículos producidos por una fábrica, etc. La Teoría Combinatoria estudia especialmente el número de agrupaciones que pueden ser obtenidas bajo algún modo de composición de los elementos. Para ello, distingue básicamente 4 conceptos: Principio fundamental de conteo, Principio de la suma, permutaciones, combinaciones. (12)

Otra definición propuesta es que la **Teoría Combinatoria** es el arte de contar, es decir, de calcular inteligentemente cardinales de conjuntos y de enumerar, esto es, determinar los elementos de un conjunto descrito por alguna propiedad. Es una disciplina clásica que cobra nuevo auge con la aparición de los ordenadores por dos razones: por un lado por la posibilidad de cálculo que éstos aportan, y por otro porque en el estudio de algoritmos o en el análisis de programas los problemas del tipo cálculo del número de operaciones, unidades de memoria que se precisan para realizar una cierta operación, estudio de la complejidad son problemas de tipo combinatorio. (13)

Principio Fundamental de Conteo

Definición 1. *Si una actividad consta de t pasos y el paso 1 puede realizarse de $n_1$formas; y una vez hecho el primer paso, el paso 2 se puede realizar de n_2 formas; y una vez hecho esto, el paso 3 se puede efectuar de n_3. . . y finalmente el paso t se puede realizar de n_t formas, entonces el número de posibles actividades a realizar es:*

$$\prod_{i=1}^{t} n_i = n_1 * n_2 * \ldots * n_t$$

Este principio es también conocido como Regla de la Multiplicación.*(14)*

Principio de la suma

Definición 2. *Sean $X_1, X_2,. . ., Xt$ conjuntos disjuntos dos a dos, es decir, si $i \neq j$ entonces $X_i \cap X_j = \emptyset$, y sea que X_i tiene ni elementos, i = 1. . . n. (14) Entonces, el número de formas en puede elegirse un elemento de X_1, o de $X_2,. . .$, o de X_t, es:*

$$\sum_{i=1}^{t} n_i = n_1 + n_2 + \cdots + n_t$$

Permutaciones

Para entender lo que son las permutaciones es necesario definir lo que es una combinación y lo que es una permutación para establecer su diferencia y de esta manera entender claramente cuando es posible utilizar una combinación y cuando utilizar una permutación al momento de querer cuantificar los elementos de algún evento.

Definición 3. Permutación: *Dado un conjunto A de n elementos distintos $x_1, x_2,. . ., x_n$, se llama permutación de los elementos de A, a una ordenación estos n elementos en una fila. (14)*

Definición 4.Combinación: *Es todo arreglo de elementos en donde no nos interesa el lugar o posición que ocupa cada uno de los elementos que constituyen dicho arreglo. (14)*

Dado un conjunto de n elementos, se llama permutación de n a cada forma de ordenar los$n$ elementos dados. Estas se clasifican de dos maneras:

- ✓ Permutaciones con repetición.
- ✓ Permutaciones sin repetición.

Permutaciones de n con repetición.

Se llaman Permutaciones de n con elementos repetidos aquellas donde el primer elemento se repite *a* veces, el segundo *b* veces, el tercero *c* veces,...

Donde *n=a + b + c+***...**

Son los distintos grupos que pueden formarse con n elementos de forma que:

- ✓ Sí entran todos los elementos.
- ✓ Sí importa el orden.
- ✓ Sí se repiten los elementos.

$$P_{n\,=\,A^{n}_{n=}}\ \frac{n!}{a!\,b!\,c!\,\ldots}$$

Permutaciones de n sin repetición.

Se llama Permutaciones de *n* elementos (*n* = *n*) a las diferentes agrupaciones de *n* elementos de forma que:

- ✓ Sí entran todos los elementos.
- ✓ Sí importa el orden.
- ✓ No se repiten los elementos.

Por consiguiente el número de permutaciones de$n\ (P_n)$ es:

$$P_{n\,=\,A^{n}_{n=}}\ \frac{n!}{(n-n)!} = \frac{n!}{0!} = n!$$

Permutaciones circulares

Es un caso particular de las Permutaciones. Se utilizan cuando los elementos se han de ordenar "en círculo", (por ejemplo, los comensales en una mesa), de modo que el primer elemento que "se sitúe" en la muestra determina el principio y el final de muestra.

$$PC_{\mathbf{n}} = (n-1)!$$

Permutaciones de r elementos en n

Permutaciones de r elementos en n sin permitir repetición

Definición 5*. Dado un conjunto A de n elementos distintos se llama r–permutación de A sin permitir repetición, r ≤ n, a una permutación de r elementos seleccionados de A. (14) Y viene dada por la fórmula:*

$$P_{n,r} = \frac{n!}{(n-r)!}$$

Permutaciones de r en n con repetición

Definición 6. *Se llaman Permutaciones permitiendo repetición de n elementos y eliges r de ellas aquellas donde hay n posibilidades para la primera elección, después hay n posibilidades para la segunda elección*, *y así.* (14)

$$P_{n,r} = n^r$$

Donde *n* es el número de elementos que puedes elegir, y eliges *r* de ellas (Se puede repetir, el orden importa).

Combinaciones

Definición 5. *Dado un conjunto de n elementos, se define combinación de n de orden* k $(k \leq n)$ *a cada subconjunto que puede formarse tomando k elementos diferentes entre los n dados.*(14)

Estas se pueden clasificar de dos formas:

- ✓ Se puede repetir: como monedas en tu bolsillo (5,5,5,10,10)
- ✓ Sin repetición: como números de lotería (2,14,15,27,30,33)

Combinaciones con repetición.

Las combinaciones con repetición de *r* elementos tomados de *n* en *n* (m ≥ n)**,** son los distintos grupos formados por *n* elementos de manera que:

- ✓ No entran todos los elementos.
- ✓ No importa el orden.

- ✓ Sí se repiten los elementos.

$$C_{n,r} = \frac{(n+r-1)!}{r!\,(n-1)!}$$

Combinaciones sin repetición.

Se llama combinaciones de *r* elementos tomados de *n* en *n* (r ≥ n) a todas las agrupaciones posibles que pueden hacerse con *n* elementos de forma que:

- ✓ No entran todos los elementos.
- ✓ No importa el orden.
- ✓ No se repiten los elementos.

$$C_{n,r} = \frac{n!}{r!\,(n-r)!}$$

1.4.2 Teoría de Números. Principales conceptos y definiciones

La **Teoría de Números** es la rama de matemáticas puras que estudia las propiedades y de las relaciones de los números. Según esta amplia definición, la teoría de números incluye gran parte de las matemáticas, en particular del análisis matemático. Sin embargo, normalmente se limita al estudio de los números enteros y, en ocasiones, a otros conjuntos de números con propiedades similares al conjunto de los enteros. La teoría de números contiene una cantidad considerable de problemas que son "fácilmente comprendidos por los no matemáticos". De forma más general, este campo estudia los problemas que surgen con el estudio de los enteros. Entre sus principales definiciones de encuentran: La divisibilidad, el algoritmo de Euclides para calcular el máximo común divisor, la factorización de los enteros como producto de números primos y las congruencias. (15)

Divisibilidad

Definición 6. *Sean a, b dos números enteros, se dice que a divide a b y se escribe a/b si existe un número entero c tal que b = a · c. También decimos que b es múltiplo de a o que a es un factor de b.* (12)

Es decir, en el caso divisible, se puede efectuar la división *b* entre *a* y se obtiene un número entero, esto es, al hacer la división entera entre *b* y *a* el resto es 0. (12)

Ejemplo: El número entero 12 divide a 60 porque 60 = 5·12 pero 12no divide a 34.

Observaciones:

- ✓ Todo número entero $a \in \mathbb{Z}$ es divisible por *a*, por *–a,* por1 y por *–1.*
- ✓ Cada divisor *d* de un número entero no nulo $a \in \mathbb{Z}$ verifica |d| ≤ |a|, donde |x| es el valor absoluto, esto es, para cada x $\in \mathbb{Z}$ se tiene |x| = x si x ≥ 0 y |x| = *–x* si x < 0.
- ✓ Un entero positivo $a \neq 1$ se dice que es un número primo si sus únicos factores positivos son *a* y 1. Si *a* tiene un factor positivo distinto de *a* ó de 1 se dice que es compuesto.

Algoritmo de Euclides para calcular el Máximo Común Divisor (mcd)

Definición 7. *Dados dos números enteros positivos a y b se define el máximo común divisor de a y b y se escribe mcd(a; b) como el mayor de los divisores comunes de a y b.* (12)

El algoritmo de Euclides es un método eficaz para calcular el máximo común divisor (**mcd**) entre dos números enteros. El algoritmo consiste en varias divisiones euclidianas sucesivas. En la primera división, se toma como dividendo el mayor de los números y como divisor el otro (se ahorra así un paso). Luego, el divisor y el resto sirven respectivamente de dividendo y divisor de la siguiente división. El proceso termina cuando se obtiene un resto nulo. El **mcd** es entonces el penúltimo resto del algoritmo.

Observación: Dados dos números enteros positivos *a* y *b*, *a* > *b* entonces *mcd*(*a; b*) = *mcd* (*b; r*) donde *r* es el resto de dividir *a* entre *b*.

Ejemplo:

Dividimos 8872 entre 125 para obtener:

$$8872 = 125 \cdot 70 + 122;$$

Lo que indica que:

$$mcd(8872; 125) = mcd(125; 122).$$

Dividimos 125 entre 122:

$$125 = 122 \cdot 1 + 3;$$

Por tanto:

$$mcd(125; 122) = mcd\ (122; 3).$$

De nuevo:

$$122 = 3 \cdot 40 + 2;$$

Y por tanto:

$$mcd(122; 3) = mcd(3; 2).$$

Otra vez:

$$3 = 2 \cdot 1 + 1;$$

Lo que significa:

$$mcd(3; 2) = mcd\ (2; 1).$$

Definición 8. *Si dos números enteros positivos a, b $\in \mathbb{Z}$ verifican que mcd (a; b) = 1 se dicen que son primos entre sí (o relativamente primos).* (12)

MínimoComúnMúltiplo (mcm)

Definición 9. *Un múltiplo común de dos enteros a,b es un entero no negativo que es divisible por ambos a y b. Si m es múltiplo común de a,b y si m divide a todo otro múltiplo común, entonces m es el mínimo común múltiplo, de a,b, al que se denotará por* ***mcm****(a, b).* (16)

Factorización de los enteros como producto de números primos

Definición 10. *Todo número entero a ≥ 2 puede escribirse de forma única (salvo posibles reordenamientos) como producto de números primos.*

$$a = p_1^{a1} \cdot p_2^{a2} \cdot p_n^{an}$$

Donde los números a_1,..., a_n son números naturales no nulos y p_1, p_2,...,p_n son números primos distintos. A la expresión de *a* de arriba se le llama factorización prima de *a* o descomposición en factores primos de *a*. (12)

Congruencias

Definición 11. *Sea n un entero positivo y sean a y b, dos enteros cualesquiera. Se dice que a es congruente con b módulo n, o que a es un resto de b módulo n y lo denotamos por*:

$$a \equiv b\ modulo(n)$$

si a y b dan el mismo resto cuando dividen n.(17)

En la Universidad de las Ciencias Informáticas (UCI) para contribuir en el proceso de enseñanza-aprendizaje se hacen uso de diferentes asistentes matemáticos; pero no existe en la asignatura de Matemática Discreta una herramienta que trabaje solamente la combinatoria y la teoría de números. En el presente trabajo se desarrolla una herramienta que asiste al trabajo con objetos discretos, con un enfoque de apoyo en la enseñanza de la Combinatoria y la Teoría de Números en la asignatura Matemática Discreta.

1.5 Metodología de desarrollo.

En la actualidad la construcción de software tiende a ser más compleja, puesto que se requieren de sistemas más grandes y potentes, los usuarios incrementan su inclinación por software sofisticados que a medida de su perfeccionamiento cambien de una versión a otra. ”El proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema de software”. Las metodologías de desarrollo de software surgen con el objetivo de minimizar el tiempo, hacen muy eficiente y reproducible el camino para obtener resultados confiables, contienen procedimientos de gestión que coordinan y guían técnicas, que determinan las herramientas necesarias para garantizar un eficaz soporte a automatizar. (18)

Al realizar un profundo estudio se puede decir que existen diferentes metodologías de desarrollo, que se pueden aplicar a diferentes proyectos, teniendo en cuenta el flujo de información con que se trabaja en el mismo. Se clasifican en tradicionales y ágiles. Dentro de las tradicionales se encuentran: Rational Unified Process (RUP), Microsoft Solutions Framework (MSF), etc. y dentro de las ágiles se tienen: Extreme Programming (XP), SCRUM, Cristal Methodologies, entre otras. A continuación se seleccionó una metodología ágil (XP) y una tradicional (RUP) para establecer una comparación entre ellas que permita definir cuál es la más adecuada para guiar el proceso de desarrollo.

1.5.1 Metodología tradicional. Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado es un proceso de Desarrollo de Software. Este a su vez es, el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Sin embargo, el proceso unificado más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. Está basado en componentes lo cual quiere decir que el sistema de software en construcción está formado por

componentes, software interconectado a partir de interfaces bien definidas. El proceso unificado utiliza el Lenguaje Unificado de Modelado (UML), para preparar todos los esquemas de un sistema de software. Los autores del Proceso Unificado de Desarrollo destacan que el proceso de software propuesto por RUP tiene tres características esenciales:

Dirigido por casos de uso: Los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo con la que el equipo de proyecto y los usuarios deben estar de acuerdo.

Iterativo e Incremental: Es práctico dividir el trabajo en partes más pequeñas o mini-proyectos. Cada mini-proyecto es una iteración que resulta en un incremento, las iteraciones hacen referencias a pasos en el flujo de trabajo y los incrementos, al crecimiento del producto. (19)

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (20)

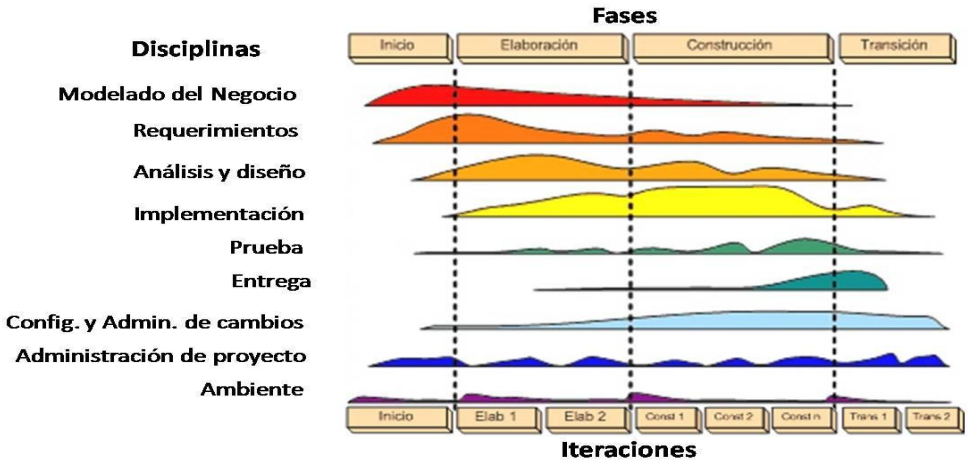


Figura 1. Proceso de desarrollo de software

1.5.2 Metodologías ágiles. Extreme Programming (XP)

XP (Extreme Programming)

Es una de las metodologías de desarrollo de software exitosas en la actualidad utilizadas para proyectos de corto plazo, que posean equipos pequeños y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. (21)

Valores de XP

XP es una disciplina de desarrollo de software basado en los valores de la simplicidad, la comunicación, la retroalimentación y coraje. Su acción consiste en llevar todo el equipo junto con la presencia de prácticas sencillas, con suficiente información para que el equipo pueda ver dónde están y ajustar las prácticas a su situación particular. (22)

Comunicación: Algunos problemas en los proyectos tienen su origen en que alguien no dijo algo importante en algún momento. XP hace casi imposible la falta de comunicación.

Simplicidad: XP propone el principio de hacer la cosa más simple que pueda funcionar, en relación con el proceso y la codificación. Es mejor hacer hoy algo simple, que hacerlo complicado y probablemente nunca usarlo mañana.

Retroalimentación: Retroalimentación concreta y frecuente del cliente, del equipo y de los usuarios finales da una mayor oportunidad de dirigir el esfuerzo eficientemente.

Coraje: El coraje (valor) existe en el contexto de los otros 3 valores. (23)

En el ciclo de vida de desarrollo de un proyecto de software los cambios van a aparecer, cambiarán los requisitos, las reglas de negocio, el personal, la tecnología, todo va a cambiar; por tanto la dificultad no es el cambio, sino la imposibilidad de adaptarnos a estos cambios. A modo de resumen XP no se basa en la planificación, el cliente forma parte del equipo de desarrollo del software, existen constantes cambios en los requisitos y es una metodología ágil apropiada para proyectos pequeños.

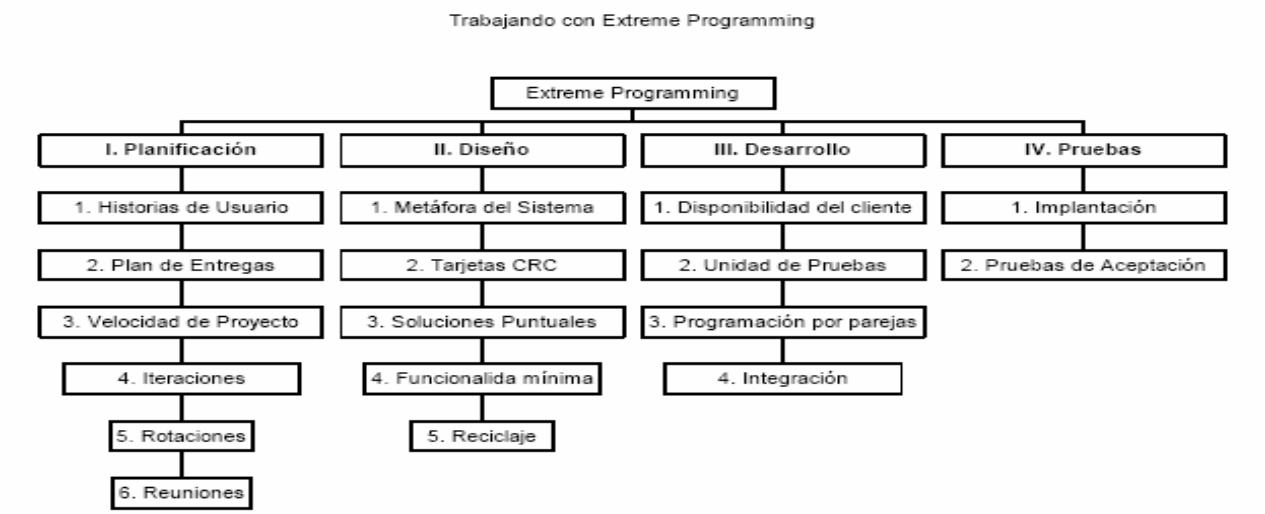


Figura 2. Representación de las fases de la Metodología XP

Se selecciona XP como metodología de desarrollo, porque se ajusta a la herramienta que se desea desarrollar, ya que el tiempo de desarrollo es un período de corta duración, el equipo de desarrollo es pequeño, donde el cliente forma parte del equipo y toma decisiones junto a éste. Esta metodología permite desarrollar un software de forma incremental y sencilla, cuyo principal objetivo es cumplir las expectativas del cliente. Presenta además una programación organizada, donde el código será revisado continuamente mediante la programación en parejas, lo que hace posible que exista una menor taza de errores. Propone además realizar pruebas, tanto unitarias, como de aceptación. Lo anteriormente planteado expresa las ventajas de la metodología seleccionada para desarrollar el software.

1.6 Herramientas y Tecnologías.

1.6.1 Lenguaje de programación. Java

El lenguaje de programación seleccionado por el equipo de desarrollo es Java debido a que es un lenguaje que permite tener desarrollos bastante completos y es de propósito general con el que se puede programar desde una aplicación Web o distribuida hasta una aplicación independiente sin ninguna conexión a Internet. Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierte en el lenguaje ideal para la creación aplicaciones distribuidas. (24)

Simple: El único requerimiento para aprender Java es tener una comprensión de los conceptos básicos de la programación orientada a objetos. Java es más complejo que un lenguaje simple, pero más sencillo

que cualquier otro lenguaje de programación. El único obstáculo que se puede presentar es conseguir comprender la programación orientada a objetos, aspecto que, al ser independiente del lenguaje, se presenta como insalvable.

Orientado a objetos: Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos.

Robusto: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Este lenguaje obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

Distribuido: Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Expandible: Al tener el código fuente disponible puedes entonces expandirlo o acoplarlo (en caso de ser necesario) a tu aplicación. Puedes heredar la funcionalidad de una clase ya existente y agregar procedimientos adicionales, posteriormente empaquetas todo y lo distribuyes. Además de la cantidad de librerías que los programadores independientes han puesto a la disposición de la comunidad utilizando esta ventaja que ofrece Java.

Dinámico: El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

1.6.2 Entorno de Desarrollo Integrado. NetBeans

NetBeans IDE es un producto libre y gratuito, sin restricciones de uso. Soporta el desarrollo de todos los tipos de aplicación Java, tales como: JEE (Java Platform Enterprise Edition), JSE (Java Standard Edition) y Java Mobile Edition. Las funciones de este IDE son provistas por módulos donde cada uno provee una función bien definida: soporte de Java, edición o soporte para el sistema de control de versiones.

Contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, facilitándole al usuario comenzar a trabajar inmediatamente. La plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Ofrece servicios comunes permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma se encuentran:

- ✓ Administración de las interfaces de usuario.
- ✓ Administración de las configuraciones del usuario.
- ✓ Administración del almacenamiento.
- ✓ Administración de ventanas.
- ✓ Marco de trabajo basado en asistentes.

Debido a que soporta todas las tecnologías definidas para el desarrollo de la solución y es multiplataforma ha sido seleccionado para la implementación del sistema.(25)

1.6.3 Herramienta Case. Visual Paradigm

Teniendo en cuenta las necesidades de la solución se decidió utilizar Visual Paradigm para el desarrollo de la misma, debido a las facilidades de esta herramienta para el trabajo colaborativo. Es una poderosa herramienta CASE que al igual que Rational Rose, utiliza UML para el modelado. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. (26)Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es una herramienta multiplataforma, fácil de instalar, actualizar y posibilita un entorno de creación de diagramas para UML 2.x. (27)

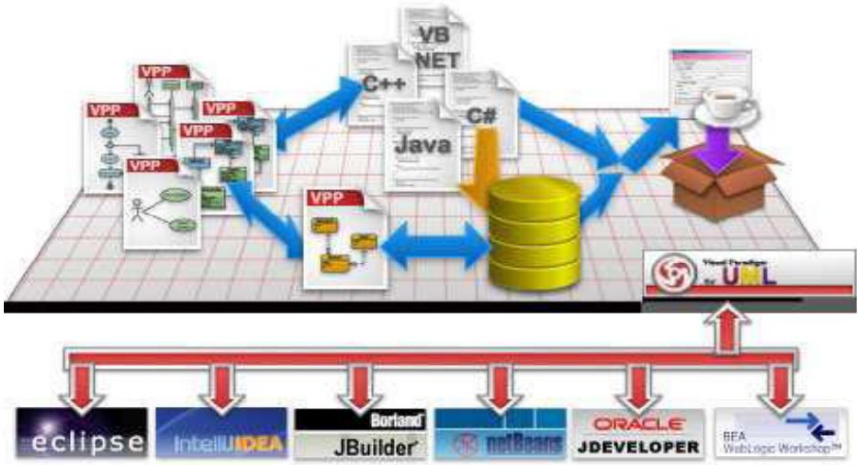


Figura 3. Integración de Visual Paradigm en el proceso de desarrollo.

1.6.4 Lenguaje de Modelado. UML

Lenguaje Unificado de Modelado (UML) es el lenguaje de modelado de sistemas más conocido y usado en la actualidad; está concebido como el estándar internacional aprobado por la Object Management Group (OMG). Se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software". (28)

UML es un conjunto de notaciones estándares destinadas a los sistemas de modelado que utilizan conceptos orientados a objetos, incluye aspectos conceptuales tales como procesos de negocio y funciones del sistema, proporciona un vocabulario y reglas que permiten la comunicación. Está compuesto por elementos que no son más que abstracciones que constituyen los bloques básicos de construcción, los cuales pueden unirse mediante relaciones para conformar los diagramas. (29)

UML es completamente independiente del lenguaje de implementación, de tal forma que los diseños realizados usando este tipo de modelado pueden ser implementados en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos); permite la automatización de determinados procesos y la generación de código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). En la elaboración de los diagramas que sean necesarios se hará uso de UML en su versión 8.0.

1.6.5 Conclusiones parciales del capítulo

En este capítulo se fundamentaron los conceptos para un mejor entendimiento en el trabajo investigativo. La metodología seleccionada permitirá el seguimiento de un proceso de desarrollo que facilite el trabajo de todos los implicados en el proyecto y la obtención de un producto con las características esperadas. El IDE NetBeans, el lenguaje de programación Java y herramienta de modelado Visual Paradigm, permitirán el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno de desarrollo bien definido.

CAPÍTULO 2: PLANIFICACIÓN Y DISEÑO DE LA HERRAMIENTA.

Introducción

Xp como metodología a utilizar guía el desarrollo del software educativo, está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. (21)

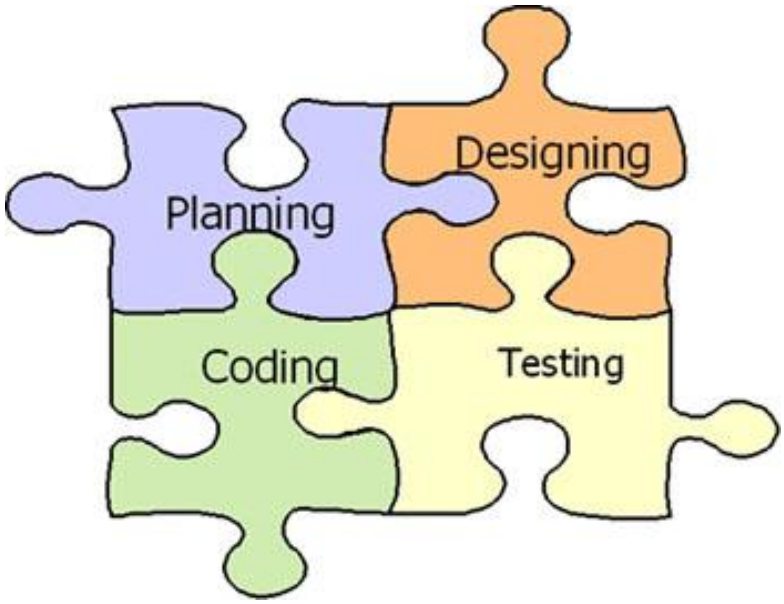


Figura 4. Programación Extrema

2.1 Fase Planificación.

La metodología de desarrollo EXtremeProgramming comienza con su fase de planificación. Durante esta etapa se realiza el proceso de identificación y desarrollo de las historias de usuario, así como la familiarización del equipo de trabajo con las tecnologías y herramientas seleccionadas para el desarrollo del software. También se acuerda el orden en que deben implementar las historias de usuario, y asociadas a éstas, las entregas, donde el resultado es un plan de entregas que contiene una estimación

de las versiones que tendrá el producto en su realización, de manera tal que guíe el desarrollo del mismo. (21)

2.1.1 Propuesta de solución.

El presente trabajo pretende desarrollar una herramienta informática, que asista al trabajo con objetos discretos, con un enfoque de apoyo al proceso de enseñanza-aprendizaje de la Combinatoria en la asignatura Matemática Discreta en la Universidad de las Ciencias Informáticas. Dicha herramienta será capaz de calcular y mostrar el número de permutaciones ya sea de n elementos o de r elementos escogidos de n, además de mostrar el listado de permutaciones que se pueden derivar, incluyendo con elementos repetidos. La herramienta permitirá además computar y visualizar el número de combinaciones de r elementos escogidos de n, además del listado de combinaciones que se pueden derivar. Incluye también funcionalidades referentes al cálculo del coeficiente binomial y el Triángulo de Pascal. Referente al tema de Teoría de Números la herramienta dará la posibilidad de realizar conversiones en los distintos sistemas numéricos, así como realizar las operaciones aritméticas básicas: suma, resta, multiplicación y división, ya sea entre números de igual base como con números de base variables, permitirá calcular el máximo común divisor y el mínimo común múltiplo de dos números enteros, además de las descomposición de factores primos y solución de Ecuaciones Lineales congruentes de la forma $ax \equiv b \mod (m)$donde m es un entero positivo, a y b son números enteros y x es una variable entera, se llama congruencia lineal o ecuación de congruencia lineal por aparecer la variable x como potencia de grado uno, únicamente y Ecuaciones Diofánticas ,estas últimas se usa para designar una ecuación en una o más incógnitas que va a ser resuelta usando los números enteros. La ecuación diofántica más simple es la ecuación diofántica lineal en dos incógnitas $ax + bx = c$, donde a y b son enteros dados, no ambos cero.

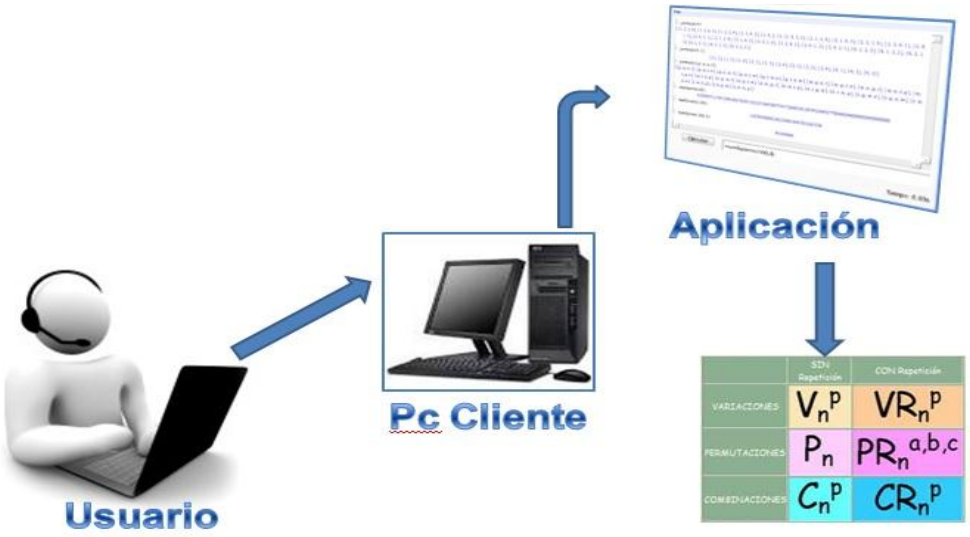


Figura 5. Propuesta de solución

Figura 6. Prototipo de Interfaz

2.1.2 Especificación de los requisitos de software

La captura y especificación de requisitos según el libro: “El proceso unificado de desarrollo de software” de Ivar Jacobson, Grady Booch y James Rumbaugh es: “el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe construir” con el propósito de “guiar el desarrollo hacia el sistema correcto” (28). Es conveniente que las ideas, necesidades o deseos de los clientes y miembros del equipo de desarrollo acerca de lo que debe hacer el sistema, sean analizadas como requisitos candidatos. Los requisitos o requerimientos se pueden clasificar en funcionales o no funcionales, siendo los funcionales las condiciones o capacidades que el sistema debe cumplir y los no funcionales las cualidades o propiedades que el producto debe tener; tratando que el mismo sea atractivo, usable, rápido y confiable entre otros.

De acuerdo a la propuesta de solución se derivan los siguientes requisitos:

Requisitos Funcionales

- ✓ **RF1 Determinar cantidad de permutaciones**

  RF1.1 Determinar la cantidad de permutaciones de n elementos.
  RF1.2 Determinar la cantidad de permutaciones posibles de r elementos de n.
  RF1.3 Determinar la cantidad de permutaciones de n elementos, *con repetición.*
  RF1.4 Determinar la cantidad de permutaciones posibles de r elementos de n, *con repetición.*

- ✓ **RF2 Generar permutaciones**

  RF2.1 Generar todas las permutaciones de un conjunto dado.
  RF2.2 Generar todas las permutaciones de un conjunto dado, *con elementos repetidos.*
  RF2.3 Generar todas las permutaciones de un conjunto dado, permitiendo *repetición de elementos.*

- ✓ **RF3 Determinar cantidad de combinaciones**

  RF3.1 Determinar la cantidad de combinaciones posibles de r elementos de n.
  RF3.2 Determinar la cantidad de combinaciones posibles de r elementos, *con repetición*, de n.

- ✓ **RF4 Generar combinaciones**

RF4.1 Generar todas las combinaciones de un conjunto dado.

RF4.2Generar todas las combinaciones con repetición de un conjunto dado.

RF4.3Generar todas las combinaciones de posibles de r elementos de n.

- ✓ **RF5 Trabajo con coeficiente binomiales.**
- ✓ **RF6 Generar Triángulo Pascal.**
- ✓ **RF7Sistemas numéricos**

RF7.1 Representar números en el sistema binario.

RF7.2 Representar números en el sistema octal.

RF7.3 Representar números en el sistema decimal.

RF7.4 Representar números en el sistema hexadecimal.

RF7.5 Realizar conversiones entre todos los sistemas numéricos.

RF7.6 Realizar operaciones aritméticas en todos los sistemas numéricos.

RF7.8 Permitir aritmética de trabajo con sistemas numéricos de *base variable*.

- ✓ **RF8 Factorizaciones**

RF8.1 Determinar el Máximo Común Divisor de dos enteros (mcd).

RF8.2 Determinar el Mínimo Común Múltiplo de dos enteros (mcm).

RF8.3 Descomponer en factores primos un número dado.

- ✓ **RF9 Determinar si dos enteros son primos relativos.**
- ✓ **RF10 Determinar si un número entero es primo.**
- ✓ **RF11 Congruencias**

RF11.1 Determinar si dos enteros son congruentes, dado un módulo.

RF11.2 Resolver ecuaciones lineales congruentes.

- ✓ **RF12 Resolver ecuaciones diofánticas.**
- ✓ **RF13 Guardar hoja de trabajo.**
- ✓ **RF14 Permitir cargar una hoja de trabajo**

Requisitos no funcionales

A continuación se presentan los requerimientos no funcionales mínimos que debe tener el sistema para su correcto funcionamiento.

- ✓ **RNF1**Portabilidad:
  - El sistema será multiplataforma, brindando la posibilidad de desplegarlo sobre diferentes Sistemas Operativos como Linux y Windows.

- ✓ **RNF2**Software:
  - Sistema Operativo Windows o Linux.
  - Instalado máquina virtual de Java preferentemente versión 6.0.x o superior.

- ✓ **RNF3**Interfaz
  - El sistema deberá contar con una interfaz de fácil entendimiento para que usuarios inexpertos puedan interactuar fácilmente con el software. Conteniendo un manual que describa el funcionamiento.

- ✓ **RNF4**Hardware
  - Para explotación del cliente: PC Pentium 3 o superior, CPU 133 MHZ o superior, 512 Mb RAM mínimo 1 Gb RAM recomendada o superior.

- ✓ **RNF5** Usabilidad

  El sistema podrá ser utilizado por cualquier usuario con las siguientes características:

  - Conocimientos básicos relativos al uso de una computadora.
  - Conocimientos básicos del sistema operativo Windows.
  - Conocimientos básicos del sistema operativo Linux.

2.1.3 Historias de Usuarios (HU)

Las Historias de Usuario son 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, sean requisitos funcionales o no funcionales, su tratamiento es muy dinámico y flexible, en cualquier momento estas pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. (21)

A continuación se presentan las HU determinadas por el cliente:

1. HU Determinar la cantidad de permutaciones de n elementos.
2. HU Generar todas las permutaciones de un conjunto dado.
3. HU Determinar la cantidad de permutaciones posibles de r elementos de n.
4. HU Determinar la cantidad de combinaciones posibles de r elementos de n.
5. HU Generar todas las combinaciones de un conjunto dado.
6. HU Determinar la cantidad de permutaciones de n elementos, *con repetición*.
7. HU Generar todas las permutaciones de un conjunto dado, *con elementos repetidos*.
8. HU Determinar la cantidad de permutaciones posibles de r elementos de n, *con repetición*.
9. HU Generar todas las permutaciones de un conjunto dado, permitiendo *repetición de elementos*.
10. HU Determinar la cantidad de combinaciones posibles de r elementos, *con repetición*, de n.
11. HU Generar todas las combinaciones con repetición de un conjunto dado.
12. HU Trabajo con coeficientes binomiales.
13. HU Generar el Triángulo de Pascal.
14. HU Representar números en el sistema binario.
15. HU Representar números en el sistema octal.
16. HU Representar números en el sistema decimal.
17. HU Representar números en el sistema hexadecimal.
18. HU Realizar conversiones entre todos los sistemas numéricos.
19. HU Realizar operaciones aritméticas en todos los sistemas numéricos.
20. HU Permitir aritmética de trabajo con sistemas numéricos de *base variable.*
21. HU Determinar el Máximo Común Divisor de dos enteros (mcd).
22. HU Determinar el Mínimo Común Múltiplo de dos enteros (mcm).

23. HU Descomponer en factores primos un número dado.
24. HU Determinar si dos enteros son primos relativos.
25. HU Determinar si un número entero es primo.
26. HU Determinar si dos enteros son congruentes, dado un módulo.
27. HU Resolver ecuaciones lineales congruentes.
28. HU Resolver ecuaciones diofánticas.
29. HU Guardar hoja de trabajo.
30. HU Permitir cargar una hoja de trabajo.

A continuación se muestran unas de las HU que poseen prioridad alta, para el cliente ver las restantes ver los Anexos.

Tabla 1. Descripción de la HU Determinar la cantidad de permutaciones de n elementos

Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Determinar la cantidad de permutaciones de n elementos.
Modificación de Historia de Usuario:	
Usuario: Usuarios	**Iteración Asignada:**1
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Media	
El usuario tiene la posibilidad de introducir un conjunto de elementos y ver la cantidad de permutaciones posibles de ese conjunto de elementos. Este conjunto de elementos pueden ser números o letras.	

Tabla 2. Descripción de la HU Generar todas las permutaciones de un conjunto dado

Historia de Usuario	
Número: 2	**Nombre de Historia de Usuario:** Generar todas las permutaciones de un conjunto dado.
Modificación de Historia de Usuario:	
Usuario: Usuarios	**Iteración Asignada:**2

Prioridad en negocio: Alta
Riesgo en Desarrollo: Media
El usuario tiene la posibilidad de introducir un conjunto de elementos y ver todas las permutaciones posibles de ese conjunto de elementos. En caso de que el conjunto sea muchos caracteres, dígase más de 7, le da posibilidad de mostrarle solo 2000 permutaciones.

Tabla 3. Descripción de la HU Determinar la cantidad de combinaciones posibles de r elementos de n

Historia de Usuario	
Número: 4	**Nombre de Historia de Usuario:** Determinar la cantidad de combinaciones posibles de r elementos de n.
Modificación de Historia de Usuario:	
Usuario: Usuarios	**Iteración Asignada:**1
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Media	
El usuario tiene la posibilidad de introducir los valores de n y r de acuerdo a la formula C(n,r), donde n puede ser un número o un conjunto de elementos, y r que es un número, ver la cantidad de combinaciones que se pueden derivar de escoger r elementos de n.	

Tabla 4. Descripción de la HU Generar todas las combinaciones de un conjunto dado

Historia de Usuario	
Número: 5	**Nombre de Historia de Usuario:** Generar todas las combinaciones de un conjunto dado.
Modificación de Historia de Usuario:	
Usuario: Usuarios	**Iteración Asignada:**2
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Media	
El usuario tiene la posibilidad de introducir un conjunto de elementos y ver todas las combinaciones posibles de ese conjunto de elementos. En caso de que el conjunto sea muchos caracteres, dígase más	

de 7, le da posibilidad de mostrarle 2000 permutaciones.

Tabla 5. Descripción de la HU Determinar la cantidad de permutaciones posibles de r elementos de n

Historia de Usuario		
Número: 3	**Nombre de Historia de Usuario:** Determinar la cantidad de permutaciones posibles de r elementos de n.	
Modificación de Historia de Usuario:		
Usuario: Usuarios		**Iteración Asignada:**1
Prioridad en negocio: Alta		
Riesgo en Desarrollo: Media		
El usuario tiene la posibilidad de introducir los valores de n y r de acuerdo a la fórmulaP(n, r), donde n puede ser un número o un conjunto de elementos, y r que es un número, ver la cantidad de permutaciones que pueden salir de escoger r elementos de n.		

2.1.4 Estimación del Esfuerzo por Historia de Usuario

En el epígrafe se realiza la estimación del esfuerzo por Historia de Usuario para lo cual se hace necesario tener en cuenta que estas deben ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, se divide en dos o más Historias. Por ejemplo las Historias de Usuarios Guardar hoja de trabajo y Cargar una hoja de trabajo, Determinar la cantidad de permutaciones de n elementos y Determinar la cantidad de permutaciones de n elementos, *con repetición,* Determinar la cantidad de permutaciones posibles de r elementos de n y Determinar la cantidad de permutaciones posibles de r elementos de n, *con repetición,* Determinar la cantidad de combinaciones posibles de r elementos de n y Determinar la cantidad de combinaciones posibles de r elementos, *con repetición*, de n, se combinan debido a que tienen una prioridad media y es considerado por parte del equipo de desarrollo que estas Historias de Usuario se realice en un período inferior a una semana.

En la tabla que se muestra a continuación los puntos de estimación representan el período de 0 a 3 semanas que se estima dure el desarrollo de las Historias de Usuarios.

Tabla 6. Puntos de Estimación

Historias de Usuarios	Puntos de Estimación
HU Determinar la cantidad de permutaciones de n elementos. HU Determinar la cantidad de permutaciones de n elementos, *con repetición.*	0.1
HU Generar todas las permutaciones de un conjunto dado.	0.3
HU Determinar la cantidad de permutaciones posibles de r elementos de n. HU Determinar la cantidad de permutaciones posibles de r elementos de n, *con repetición.*	0.1
HU Determinar la cantidad de combinaciones posibles de r elementos de n. HU Determinar la cantidad de combinaciones posibles de r elementos, *con repetición*, de n.	0.1
HU Generar todas las combinaciones de un conjunto dado.	0.3
HU Generar todas las permutaciones de un conjunto dado, *con elementos repetidos.*	0.3
HU Generar todas las permutaciones de un conjunto dado, permitiendo *repetición de elementos.*	0.3
HU Generar todas las combinaciones con repetición de un conjunto dado.	0.3
HU Trabajo con coeficientes binomiales.	0.1
HU Generar el Triángulo de Pascal.	0.3

HU Representar números en el sistema binario. HU Representar números en el sistema octal. HU Representar números en el sistema decimal. HU Representar números en el sistema hexadecimal. HU Realizar conversiones entre todos los sistemas numéricos.	0.1
HU Realizar operaciones aritméticas en todos los sistemas numéricos. HU Permitir aritmética de trabajo con sistemas numéricos de *base variable.*	0.1
HU Determinar el Máximo Común Divisor de dos enteros (mcd).	0.3
HU Determinar el Mínimo Común Múltiplo de dos enteros (mcm). HU Determinar si dos enteros son primos relativos. HU Determinar si un número entero es primo. HU Determinar si dos enteros son congruentes, dado un módulo.	0.1
HU Descomponer en factores primos un número dado.	0.3
HU Resolver ecuaciones lineales congruentes.	1
HU Resolver ecuaciones diofánticas.	1
HU Guardar hoja de trabajo y HU Permitir Cargar una hoja de trabajo	1

2.1.5 Plan de Iteraciones

A continuación se realiza el plan de iteraciones, que consiste en seleccionar las Historias de Usuario que se implementan en cada iteración, para ello se tuvo en cuenta el nivel de prioridad de las mismas. Cada iteración corresponde a un período de tiempo de desarrollo del proyecto de entre una y tres semanas.

Iteración 1

En esta iteración se implementan las Historias de Usuario 1, 3, 4, 6, 8, 10 y 12. De las cuales las HU 1, 2, 3, 4 y la 5 poseen prioridad alta, mientras el resto posee prioridad media. De suma importancia ya que permitirán mostrar la información acerca de la cantidad de permutaciones ya sea de n elementos o de r elementos escogidos n elementos P(n, r), así como mostrar la información acerca de la cantidad de combinaciones de r elementos escogidos de n C(n, r), ambas con elementos repetidos o no, incluye también la funcionalidad para el trabajo con coeficientes binomiales, que son la base fundamental del resto de las funcionalidades.

Iteración 2

En esta iteración se implementa las Historias de Usuarios 2, 5, 7, 9, 11 y 13 que tiene una alta prioridad y permitirá mostrar o listar el conjunto de permutaciones de n elementos o de r elementos escogidos de n elementos, ya sea con repetición o sin repetición, dando la posibilidad de mostrar y listar este conjunto de permutaciones con permutaciones repetidas o no. Permitirá mostrar y listar el conjunto de combinaciones de un conjunto de elementos o de r elementos escogidos de n elementos, ya sea con repetición o sin repetición, así como mostrar el triángulo de Pascal a partir de un número entero n.

Iteración 3

En esta iteración se implementa las Historias de Usuarios 14, 15, 16, 17, 18, 19, y 20, de las cuales 14, 15, 16,17 y 20 tienen prioridad alta y el resto prioridad media. Permitirá mostrar las conversiones que se pueden realizar en los diferentes sistemas números, así como las operaciones aritméticas que se pueden realizar, ya sea entre números de un mismo sistema o números expresados en sistemas diferentes.

Iteración 4

En esta iteración se implementa las Historias de Usuarios 21, 22, 23, 24, 25, 26, 27, 28, 29 y 30, de las cuales las HU 21, 27 y 28 tienen prioridad alta y el resto prioridad media. En esta iteración se implementan las Historias de Usuarios que dan la posibilidad que calcular el mínimo común múltiplo y el máximo común divisor de dos números, además de mostrar el listado de factores primos que se derivan de la descomposición en factores de un número. Permitirá saber si dos números enteros son primos relativos, y

dado un módulo si son congruentes, además de resolver ecuaciones lineales congruentes y ecuaciones diofánticas.

Tabla 7. Plan de duración de las Iteraciones

Iteraciones	Historias de Usuario(HU)	Duración Total
Iteración 1	1. HU Determinar la cantidad de permutaciones de n elementos. 2. HU Determinar la cantidad de permutaciones posibles de r elementos de n. 3. HU Determinar la cantidad de combinaciones posibles de r elementos de n. 4. HU Determinar la cantidad de permutaciones de n elementos, *con repetición.* 5. HU Determinar la cantidad de permutaciones posibles de r elementos de n, *con repetición.* 6. HU Determinar la cantidad de combinaciones posibles de r elementos, *con repetición*, de n. 7. HU Trabajo con coeficientes binomiales.	0.4 semana
Iteración 2	1. HU Generar todas las permutaciones de un conjunto dado. 2. HU Generar todas las permutaciones de un conjunto dado, *con elementos repetidos.* 3. HU Generar todas las permutaciones de un conjunto dado, permitiendo *repetición de elementos.* 4. HU Generar el Triángulo de Pascal. 5. HU Generar todas las combinaciones de un conjunto dado. 6. HU Generar la cantidad de combinaciones posibles de r elementos, *con repetición*, de n. 7. HU Generar todas las combinaciones con repetición de un conjunto dado.	3 semanas
Iteración 3	1. HU Representar números en el sistema binario. 2. HU Representar números en el sistema octal. 3. HU Representar números en el sistema decimal.	0.2 semanas

	4. HU Representar números en el sistema hexadecimal. 5. HU Realizar conversiones entre todos los sistemas numéricos. 6. HU Realizar operaciones aritméticas en todos los sistemas numéricos. **7.** HU Permitir aritmética de trabajo con sistemas numéricos de *base variable.*	
Iteración 4	1. HU Determinar el Máximo Común Divisor de dos enteros (mcd). 2. HU Determinar el Mínimo Común Múltiplo de dos enteros (mcm). 3. HU Descomponer en factores primos un número dado. 4. HU Determinar si dos enteros son primos relativos. 5. HU Determinar si un número entero es primo. 6. HU Determinar si dos enteros son congruentes, dado un módulo. 7. HU Resolver ecuaciones lineales congruentes. 8. HU Resolver ecuaciones diofánticas. 9. HU Guardar hoja de trabajo. 10. HU Permitir cargar una hoja de trabajo.	4 semanas

2.1.6 Plan de Entrega

Con las historias de usuario se crea el plan estimado de entrega. El cronograma de entregas establece qué historias de usuarios se agrupan para conformar una entrega y el orden de las mismas. Típicamente el cliente ordena y agrupa según sus prioridades las historias de usuarios.

En este plan se decide realizar cuatro entregas, las cuales se reflejan en la tabla que se muestra a continuación.

Tabla 8. Plan de Entregas

Entregas	Historias de Usuarios
Entrega 1	HU Determinar la cantidad de permutaciones de n elementos.
	HU Determinar la cantidad de permutaciones posibles de r elementos de n.
	HU Determinar la cantidad de combinaciones posibles de r elementos de n.
	HU Determinar la cantidad de permutaciones de n elementos, *con repetición.*

	HU Determinar la cantidad de permutaciones posibles de r elementos de n, *con repetición.*
	HU Determinar la cantidad de combinaciones posibles de r elementos, *con repetición*, de n.
	HU Trabajo con coeficientes binomiales.
Entrega 2	HU Generar todas las permutaciones de un conjunto dado.
	HU Generar todas las permutaciones de un conjunto dado, *con elementos repetidos.*
	HU Generar todas las permutaciones de un conjunto dado, permitiendo *repetición de elementos.*
	HU Generar todas las combinaciones de un conjunto dado.
	HU Generar todas las combinaciones con repetición de un conjunto dado.
	HU Generar el Triángulo de Pascal.
Entrega 3	HU Representar números en el sistema binario.
	HU Representar números en el sistema octal.
	HU Representar números en el sistema decimal.
	HU Representar números en el sistema hexadecimal.
	HU Realizar conversiones entre todos los sistemas numéricos.
	HU Realizar operaciones aritméticas en todos los sistemas numéricos.
	HU Permitir aritmética de trabajo con sistemas numéricos de *base variable.*
Entrega 4	HU Determinar el Máximo Común Divisor de dos enteros (mcd).
	HU Determinar el Mínimo Común Múltiplo de dos enteros (mcm).
	HU Descomponer en factores primos un número dado.
	HU Determinar si dos enteros son primos relativos.
	HU Determinar si un número entero es primo.
	HU Determinar si dos enteros son congruentes, dado un módulo.
	HU Resolver ecuaciones lineales congruentes.
	HU Resolver ecuaciones diofánticas.
	HU Guardar hoja de trabajo.
	HU Permitir cargar una hoja de trabajo.

2.2 Fase de Diseño

En esta fase XP establece una serie de recomendaciones o premisas, algunas de ellas son:

Elegir una metáfora para el sistema

En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo, que no es más que una historia compartida que describe cómo debería funcionar el sistema. Lo que permite mantener la coherencia de nombres de todo aquello que se va a implementar. Es algo que todos entienden sin necesidad de mayores explicaciones, como una manera sencilla de explicar el propósito del proyecto y guiar la estructura y arquitectura del mismo. (21)

Por ejemplo, puede ser una guía para la nomenclatura de los métodos y las clases utilizadas en el diseño del código. Tener nombres claros, que no requieran de mayores explicaciones, posibilita el ahorro de tiempo y facilita la comprensión del código.

La metáfora que se selecciona para el desarrollo de la aplicación es utilizar el vocabulario del negocio, es decir, las funciones o comandos que se utilizará en la hoja de cálculo, las clases se llaman según la funcionalidad que implementan, por ejemplo, para calcular o determinar la cantidad de permutaciones, se implementar clase CalculadorValores, donde contendrá adentro todos los métodos referente al cálculo de la cantidad permutaciones o combinaciones que se pueden derivar de un conjunto, etc.

Estándares de programación

XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios. (21)

En el caso del software que se desarrolla el estándar que se emplea en la implementación del producto es definido por los desarrolladores en conjunto con el cliente. Por ejemplo el nombre de las clases será en

mayúsculas y en caso de que el nombre sea compuesto, será en mayúscula unidos, ejemplo: CalculadorValores, CalculadorConjuntos etc. Los métodos de las clases al igual que las clases, se escribirán en mayúscula y caso de que sea un nombre compuesto será escrito unido, ejemplo: RPermutaciones, RCombinaciones, etc. Todos los métodos deberán estar comentariados en su parte superior, especificando el objetivo del mismo.

40 horas por semana

Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo. Los proyectos que requieren trabajo extra para intentar cumplir con los plazos suelen al final ser entregados con retraso. En lugar de esto se puede realizar el juego de la planificación para cambiar el ámbito del proyecto o la fecha de entrega. (21)

Entregas pequeñas

La idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendidas para el sistema que sí constituyen un resultado de valor para el negocio, donde una entrega no debería tardar más 3 meses. (21)

2.2.1 Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración)

Una de las recomendaciones más importante de la fase de diseño es la elaboración de las tarjetas CRC. El uso de estas permite al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Sirven para diseñar el sistema en conjunto entre todo el equipo. (29)

A continuación se muestran las tarjetas CRC de las clases más significativas de la herramienta, el resto el cliente podrá verlas en los Anexos: CalculadorValores, CalculadorConjuntos, CalculadorEnteros, ConversorNúmeros, Comunes.

Tabla 9. Clase: CalculadorValores

Responsabilidades	Clases Relacionadas
En esta clase se declaran los atributos que contienen la información referente al cálculo de valores, dígase cálculo de cantidad de	Valores, Comunes

permutaciones o de combinaciones, de n elementos o de r elementos escogidos de n, así como los métodos que permiten acceder y transformar la misma.	

Tabla 10. Clase: CalculadorConjuntos

Responsabilidades	Clases Relacionadas
En esta clase se declaran los atributos que contienen la información referente al cálculo de conjuntos, dígase generar o listar las permutaciones o combinaciones, de n elementos o de r elementos escogidos de n, así como los métodos que permiten acceder y transformar la misma.	Conjuntos, Comunes

Tabla 11. Clase: CalculadorEnteros

Responsabilidades	Clases Relacionadas
En esta clase se declaran los atributos que contienen la información referente al cálculo de números enteros, dígase comprobar si un número es primo, si dos enteros son congruentes, el cálculo de mcd y mcm de dos enteros, si dos enteros son relativos, así como los métodos que permiten acceder y transformar la misma.	Enteros, Comunes

Tabla 12. Clase: ConversorNúmeros

Responsabilidades	Clases Relacionadas

En esta clase se declaran los atributos que contienen la información referente a la conversión de sistemas numéricos, dígase convertir un número expresado en cierto sistema numérico a otro sistema numérico escogido por el usuario, y viceversa, así como los métodos que permiten acceder y transformar la misma.	Conversor, Comunes

Para un mejor entendimiento de las relaciones de clases se propone el diagrama de clase que se muestra a continuación.

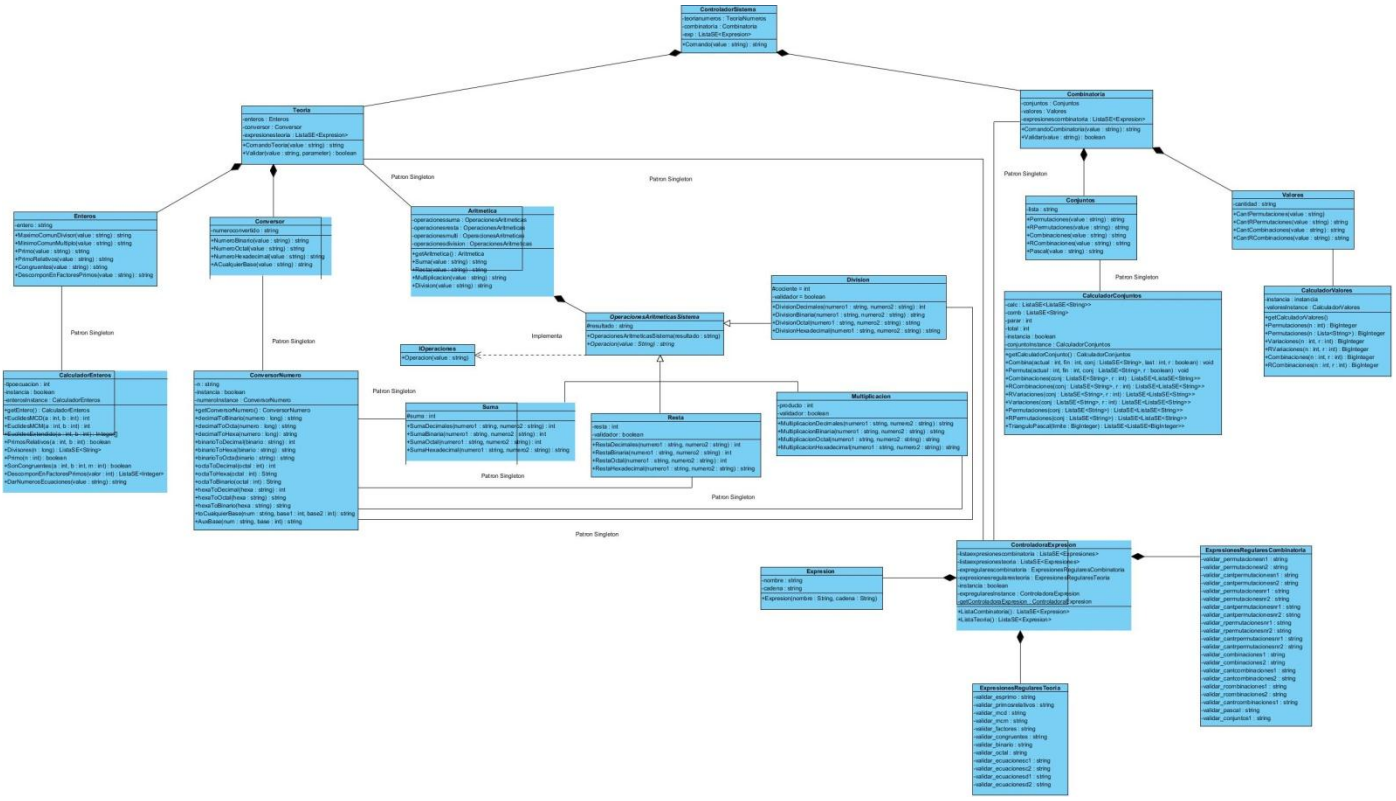


Figura 7. Diagrama de Clases del Sistema

2.3 Conclusiones parciales del capítulo

En el capítulo se documentan las historias de usuarios realizadas durante la fase de planificación por el cliente, determinándose las funcionalidades que debe cumplir el producto. Se realiza la estimación del esfuerzo que requiere cada historia de usuario a la hora de la implementación, se construye el plan de iteraciones, se realiza la estimación del tiempo que requiere la implementación de cada iteración y se lleva a cabo el plan de entregas en el cual se define qué historias de usuarios iban a formar parte de cada entrega.

CAPÍTULO 3: DESARROLLO Y PRUEBAS DE LA HERRAMIENTA

Introducción

En el capítulo se realiza la documentación referente a las fases de desarrollo y prueba según la metodología XP. Para esto se lleva a cabo la descripción de algunas de las características que se deben tener en cuenta en la fase de desarrollo. Se desglosan los requisitos funcionales en tareas con el objetivo de facilitar el trabajo de implementación. Y en la fase de pruebas se muestran los resultados de las pruebas de aceptación realizadas a la aplicación.

3.1 Fase de Desarrollo

En la fase de desarrollo XP se propone tener en cuenta una serie de características o cualidades para el correcto tratamiento del software, las que se consideran fundamentales para obtener resultados satisfactorios en la implementación del mismo.

Cliente in-situ

El cliente siempre estuvo disponible, pues no solo ayudó al equipo de desarrollo sino que formó parte de él, guiando la toma de decisiones y plasmando la descripción de las historias de usuarios, decidió si las versiones del producto resultaban de su agrado y si se cumplían los objetivos trazados, lo que contribuyó en la organización de los contenidos y guió la estrategia pedagógica que persigue la estructuración de los temas en la aplicación. (21)

Programación en parejas

La programación en pareja es una de las particularidades que propone la metodología XP para la fase de implementación, esta fue cumplida por parte del equipo de desarrollo ya que todo el código realizado se desarrolló por dos personas que trabajaron de forma conjunta en un ordenador, logrando incrementar la calidad del software desarrollado sin afectar el tiempo de entrega, lo que proporciona facilidad, seguridad y calidad en la implementación. (21)

3.1.1 Arquitectura del Sistema

El Instituto de Ingenieros Eléctricos y Electrónicos conocido por sus siglas en inglés como IEEE define la Arquitectura del Software en el estándar Std 1471-2000, como: “la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”. (31)

En el libro "An Introduction to Software Architecture", David Garlan y Mary Shaw definen que la Arquitectura es un nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema". (32) De forma general la arquitectura de software alude a —la estructura general del software y las formas en que la estructura proporciona una integridad conceptual para un sistema. En su forma más simple, la arquitectura es la estructura u organización de los componentes del programa (módulos), la manera en que estos componentes interactúan, y la estructura de datos que utilizan los componentes. En sentido más amplio, sin embargo, los componentes pueden generalizarse para representar elementos importantes del sistema y sus interacciones. (33).

Arquitectura N-Capas(N-Layer)

Para dar respuesta a los requerimientos la arquitectura estructural que posee la herramienta está sustentada por el estilo arquitectónico en **capas**, esta se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades las funcionalidades que implementan. N capas es un estilo arquitectónico donde cada capa puede ser modificada tanto como sea necesario sin provocar cambios en las demás, de no poseer dependencias con la capa superior. Esto permite el desarrollo de aplicaciones más robustas debido al encapsulamiento. Es factible un mantenimiento y soporte más sencillo así como una mayor flexibilidad. (34)La aplicación estará estructurada por 3-capas: Capa de Interfaz, Capa de Lógica de Negocio y Capa de Persistencia.

Figura 8. Arquitectura N-Capas

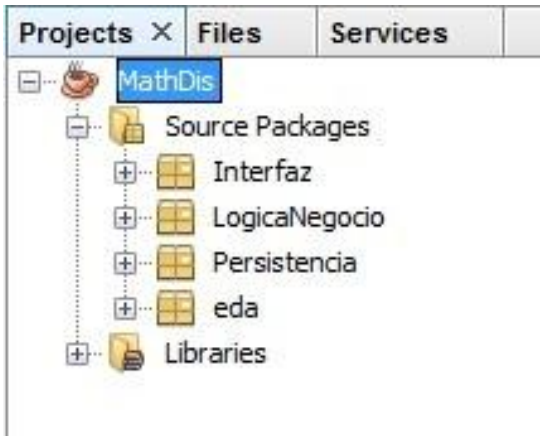


Figura 9. Arquitectura de la herramienta

- ✓ La capa de presentación contiene los formularios con los que interactúan los usuarios finales del sistema y permiten la visualización de toda la información que estos necesitan y las acciones que gestionan las funcionalidades de los formularios. Básicamente, se responsabiliza de que se le comunique información al usuario por parte del sistema y viceversa, manteniendo una comunicación exclusiva con la capa de negocio.
- ✓ La capa de negocio es la capa que contiene los procesos a realizar con la información recibida desde la capa de presentación responsabilizándose además de que se le envíen las respuestas adecuadas a la capa de presentación contiene la mayor parte de la implementación de la lógica de negocio. Responsable de las tareas propias de la aplicación, implementa la lógica de la aplicación, aplica las reglas de negocio y las entradas del usuario, es donde se procesan los datos matemáticos.
- ✓ La capa de persistencia contiene las clases persistentes para la gestión de los datos. Es responsable de la gestión y almacenamiento permanente de los datos, que en este caso los datos no se guardaran en un servidor de datos convencional, sino que cuando sea guardar la hoja de cálculo se hará en la PC donde se esté trabajando.

3.1.2 Patrones de Diseño

Un patrón de diseño es una solución estándar para un problema común de programación, una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios, una estructura de implementación que logra una finalidad determinada, manera práctica de describir aspectos de la organización de un

programa. Brindan la facilidad de reutilizar el conocimiento de desarrolladores, clasificando y describiendo problemas y soluciones a problemas que surgen con frecuencia durante el desarrollo, por lo que se convierten en un historial que no ayuda a no cometer errores ya descritos. (35)

La finalidad de cada patrón de diseño es proporcionar una descripción que le permita determinar al diseñador:

- ✓ Si el patrón es aplicable al trabajo actual.
- ✓ Si el patrón se puede reutilizar.
- ✓ Si el patrón puede servir como guía para desarrollar un patrón similar, pero diferente en cuanto a la funcionalidad o estructura.

¿Por qué surgen los patrones de diseño? Por la necesidad de transmitir la experiencia.

Con el conocimiento de estos patrones, los programadores expertos son capaces de identificar las situaciones en las que éstos tienen aplicación, y utilizarlos sin tener que detenerse para analizar el problema y vislumbrar diferentes estrategias de resolución.

Los patrones más importantes a usar en el desarrollo:

Patrones Gof

Patrón Singleton: El Singleton Pattern, es un patrón creacional, o de creación, quea diferencia de otros patrones creacionales, no se encarga de la creación de objetos en sí, sino que se enfoca en la restricción en la creación de un objeto. Siempre que se crea un objeto nuevo (en Java con la palabra reservada new) se invoca al constructor del objeto para que cree una instancia. Por lo general los constructores son públicos. El Singleton lo que hace es convertir al constructor en privado, de manera que nadie lo pueda instanciar.(36)Lo que proporciona un punto de acceso global a dichas clases. Reduce el espacio de nombres, es una mejora sobre las variables globales. (37) Es usado debido a la necesidad de trabajar con el mismo objeto en distintos momentos y distintos módulos. Las clases CalculadorConjuntos, CalculadorValores, Combinatoria, CalculadorEnteros, Aritmética, Combinatoria, ControladoraExpresion, Comunes, ConversorNumeros, Suma, Resta, División y Multiplicación utilizan este patrón.

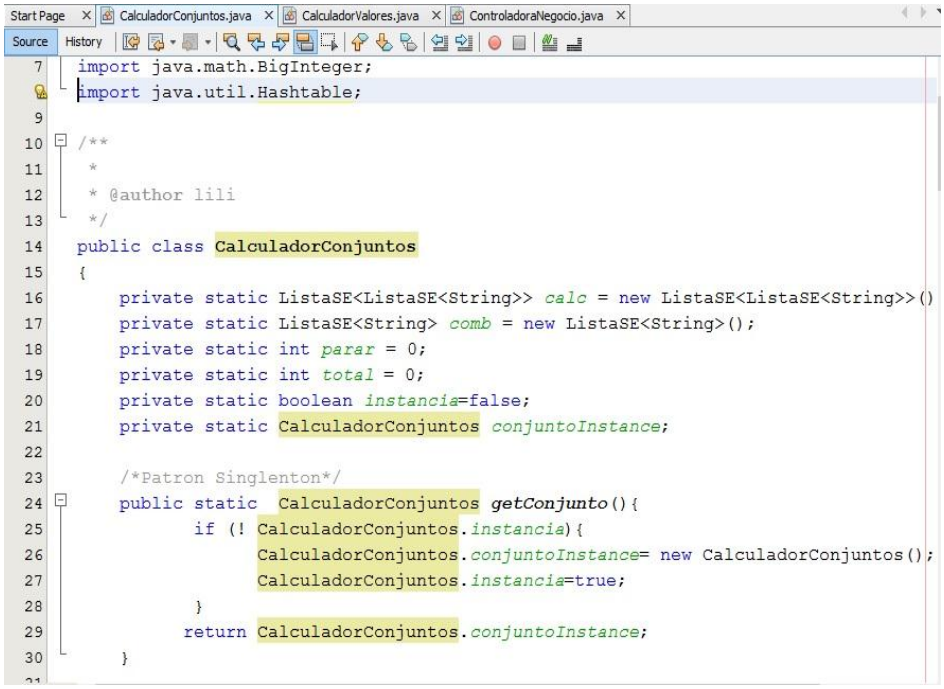


Figura 10. Ejemplo patrón Singleton en la Clase CalculadorConjuntos

En cualquier lugar de la aplicación que se quiera utilizar algún método de esta clase se realiza de la manera que se muestra a continuación:

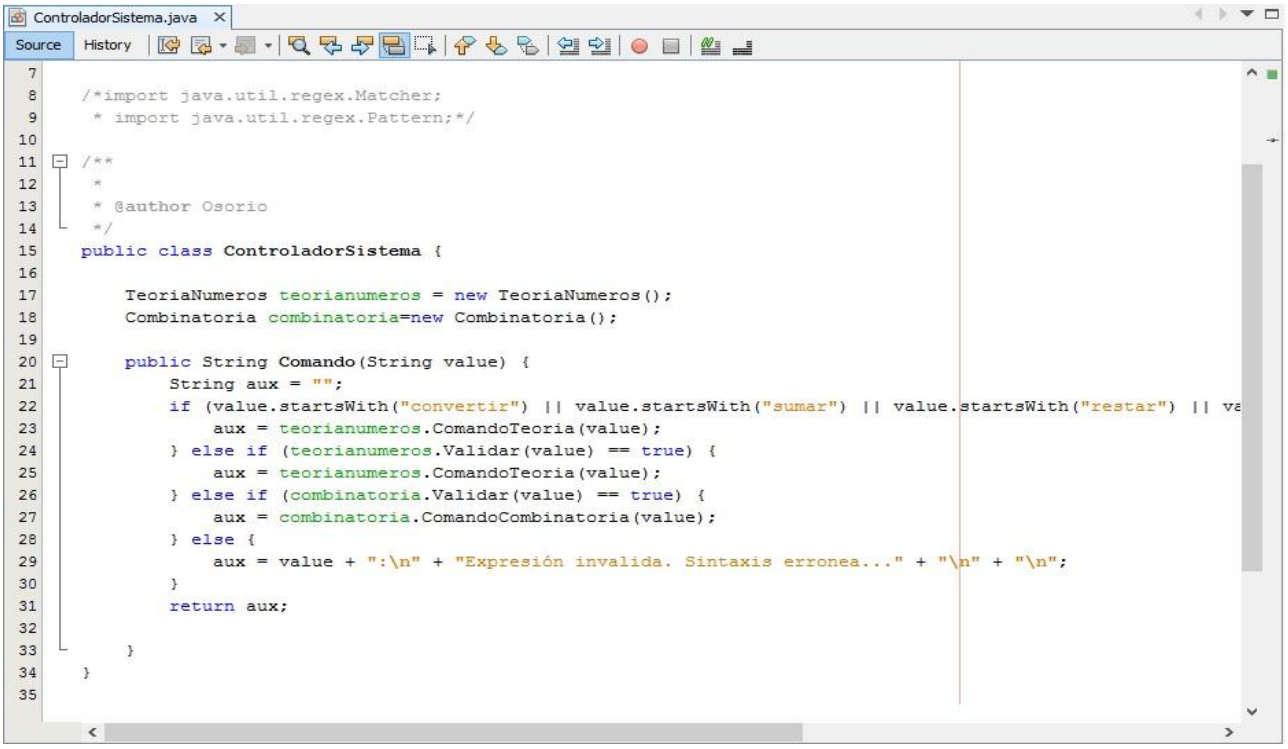
```
if(result.longitud<7)
{
    rr = CalculadorConjuntos.getConjunto().Permutaciones(result, 0);
    for (int i = 0; i < rr.longitud; i++)
    {
```

Figura 11. Patrón Singleton

Patrones Grasp

Los **patrones GRASP** son patrones generales de software para la asignación de responsabilidades a objetos. Describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades. (37)

- ✓ **Controlador:** Es utilizado como intermediario entre las interfaces y el algoritmo que las implementa. Este patrón sugiere que la lógica del negocio este separada de la capa de presentación ya que esto posibilita una mayor reutilización del código y un mayor control.




```
7
8   /*import java.util.regex.Matcher;
9    * import java.util.regex.Pattern;*/
10
11  /**
12   *
13   * @author Osorio
14   */
15  public class ControladorSistema {
16
17      TeoriaNumeros teorianumeros = new TeoriaNumeros();
18      Combinatoria combinatoria=new Combinatoria();
19
20      public String Comando(String value) {
21          String aux = "";
22          if (value.startsWith("convertir") || value.startsWith("sumar") || value.startsWith("restar") || va
23              aux = teorianumeros.ComandoTeoria(value);
24          } else if (teorianumeros.Validar(value) == true) {
25              aux = teorianumeros.ComandoTeoria(value);
26          } else if (combinatoria.Validar(value) == true) {
27              aux = combinatoria.ComandoCombinatoria(value);
28          } else {
29              aux = value + ":\n" + "Expresión invalida. Sintaxis erronea..." + "\n" + "\n";
30          }
31          return aux;
32
33      }
34  }
35
```

Figura 12. Ejemplo patrón Controlador en la Clase ControladorSistema

- ✓ **Experto**: Este patrón nos indica que la implementación de un determinado método o la creación de un objeto debe recaer sobre la clase que conoce todo la información para crearlo.
- ✓ **Creador**: Este patrón como su nombre lo indica es el que crea, el guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

1. B contiene A.
2. B es una agregación (o composición) de A
3. B almacena A
4. B tiene los datos de inicialización de A (datos que requiere su constructor)
5. B usa A.

A la hora de crear objetos se deben tener en cuenta las características de la clase.

- ✓ **Polimorfismo:** Este patrón significa “asignar el mismo nombre a servicios en varios objetos”, cuando los servicios se parecen o están relacionados entre sí. Cuando por el tipo varían las alternativas o comportamientos afines, las responsabilidades del comportamiento se asignarán - mediante operaciones polimórficas - a los tipos en que el comportamiento presenta variantes.
- ✓ **Alta Cohesión**: Este patrón es utilizado para evita asignar demasiadas responsabilidades a las clases. Brinda la posibilidad de que la información que se almacene en cada clase sea coherente. Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- ✓ **Bajo Acoplamiento**: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo o débil acoplamiento no depende de muchas otras.

  Una clase con alto o fuerte acoplamiento recurre a muchas otras, este tipo de clases no es conveniente ya que puede presentar los siguientes problemas:

  - ✓ Los cambios de las clases afines, provocan cambios locales.
  - ✓ Son más difíciles de entender cuando están aisladas.
  - ✓ Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

3.1.3 Tareas

En esta fase se hace necesario desglosar los requisitos funcionales en tareas con el objetivo de facilitar el trabajo, por lo que durante el transcurso de cada una de las iteraciones se planifican las tareas a realizar para dar solución a las historias de usuario que se derivan de estos. Después de realizar una revisión del plan de iteraciones, se empiezan a descomponer los requisitos funcionales en tareas, asignando 2 personas como responsables del desarrollo de cada una de estas, donde cada tarea corresponde a un período de uno a tres días de desarrollo.

En la iteración 1 se desarrollan las HU de los requisitos funcionales que poseen prioridad media para el cliente y a su vez para el grupo de desarrollo, no son consideradas críticas, pues se pueden implementar

en un período de tiempo menor con respecto a las restantes HU de alta prioridad, por lo que se decide descomponer de la manera que se muestra en la tabla que a continuación se presenta.

Tabla 13. Tareas de la Iteración 1

Requisito Funcional	Tareas
Determinar cantidad de permutaciones	Crear funcionalidad para determinar la cantidad de permutaciones de n elementos.
	Crear funcionalidad para determinar la cantidad de permutaciones posibles de r elementos de n.
	Crear funcionalidad para determinar la cantidad de permutaciones de n elementos, *con repetición.*
	Crear funcionalidad para determinar la cantidad de permutaciones posibles de r elementos de n, *con repetición.*
Determinar cantidad de combinaciones	Crear funcionalidad para determinar la cantidad de combinaciones posibles de r elementos de n.
	Crear funcionalidad para determinar la cantidad de combinaciones posibles de r elementos, *con repetición*, de n.
Trabajo con coeficiente binomiales	Crear funcionalidad para determinar coeficiente binomial.

En la iteración 2 se realiza las historias de usuarios de los requisitos funcionales Generar permutaciones y Generar combinaciones de alta prioridad y se estimó que requiere de 3 semanas para su realización ya que es la encargada de mostrar las permutaciones o combinaciones, de n elementos, o de r elementos escogidos de n elementos.

Tabla 14. Tareas de la Iteración 2

Requisito Funcional	Tarea
Generar permutaciones	Crear funcionalidad para generar todas las

	permutaciones de un conjunto dado.
	Crear funcionalidad para generar todas las permutaciones de un conjunto dado, *con elementos repetidos.*
	Crear funcionalidad para generar todas las permutaciones de un conjunto dado, permitiendo *repetición de elementos.*
Generar Combinaciones	Crear funcionalidad para generar todas las combinaciones de un conjunto dado.
	Crear funcionalidad para generar combinaciones posibles de r elementos, *con repetición*, de n.
Generar Triangulo Pascal	Crear funcionalidad para generar Triangulo Pascal.

En la iteración 3 se agruparon las HU de los requisitos funcionales referente al trabajo con los diferentes sistemas numéricos, ya sea conversiones de un sistema a otro o aritmética números de un mismo sistema o diferentes sistemas.

Tabla 15. Tareas de la Iteración 3

Requisito Funcional	Tarea
Sistemas Numéricos	HU Representar números en el sistema binario.
	HU Representar números en el sistema octal.
	HU Representar números en el sistema decimal.
	HU Representar números en el sistema hexadecimal.
	HU Realizar conversiones entre todos los sistemas numéricos.
	HU Realizar operaciones aritméticas en todos los sistemas numéricos.

	HU Permitir aritmética de trabajo con sistemas numéricos de *base variable.*

En la iteración 4 se agruparon todas las HU de los requisitos funcionales referente al tema de Factorizaciones y de prioridad media Guardar y Cargar una hoja de trabajo, las cuales requieren de un tiempo menor de realización. A continuación, en la tabla, se muestra la forma en que se desglosan cada una de estas.

Tabla 16. Tareas de la Iteración 4

Requisito Funcional	Tarea
Factorizaciones	HU Determinar el Máximo Común Divisor de dos enteros (mcd).
	HU Determinar el Mínimo Común Múltiplo de dos enteros (mcm).
	HU Descomponer en factores primos un número dado.
	HU Determinar si dos enteros son primos relativos.
	HU Determinar si un número entero es primo.
	HU Determinar si dos enteros son congruentes, dado un módulo.
	HU Resolver ecuaciones lineales congruentes.
	HU Resolver ecuaciones diofánticas.
Guardar una hoja de trabajo	Crear clase que permita guardar la hoja de trabajo de la herramienta.
Cargar una hoja de trabajo	Crear clase que permita cargar la hoja de trabajo de la herramienta.

Las tablas que se presentan posteriormente son las descripciones de las tareas de las HU de alta prioridad, se muestra la fecha de inicio y de fin de cada de una de ellas, así como la descripción de las

mismas. También se muestran los puntos estimados como resultado de dividir los días de duración de la tarea entre los puntos de estimación de la HU a la que pertenece y los responsables de la realización de estas.

Tabla 17. Tarea 1: Crear funcionalidad Determinar cantidad de permutaciones de n elementos

Tarea	
Númerode Tarea: 1	Requisito Funcional No: 1.1Determinar cantidad de permutaciones.
Nombre de Tarea: Crear funcionalidad para determinar la cantidad de permutaciones de n elementos.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 26/03/2013	Fecha Fin: 27/03/2013
Programador Responsable: Jorge Osorio y Lianet García Ricardo	
Descripción: Implementar funcionalidad que dado un conjunto n o un número n de la cantidad de permutaciones que se derivan.	

La última fase que propone XP es la de pruebas esta constituye uno de los cimientos básicos de la metodología, pues guía a los desarrolladores a realizar pruebas continuas al sistema logrando de esta forma que la probabilidad de que exista errores o fallos en la aplicación sea mínima.

3.2 Fase de Pruebas

La verificación y validación del software se define como actividades que se realizan para comprobar si las salidas en las diferentes etapas y/o subprocesos de desarrollo (código y artefactos) cumplen con las condiciones o los requerimientos impuestos sobre ellos en las entradas por las etapas previas.(38)

La verificación y validación del software se define como actividades que se realizan para comprobar si las salidas en las diferentes etapas y/o subprocesos de desarrollo (código y artefactos) cumplen con las condiciones o los requerimientos impuestos sobre ellos en las entradas por las etapas previas. Constatan que el producto de cada etapa del proyecto es adecuado, completo, consistente y que está acorde a los requerimientos establecidos para el software en las entradas. Esto quiere decir que con las verificaciones se asegura que el producto software satisface los requisitos especificados para el diseño en el transcurso de todas las actividades realizadas durante el ciclo de vida del desarrollo. (39)

XP propone la realización de dos tipos de pruebas al sistema que son las llamadas pruebas unitarias y pruebas de aceptación. Las pruebas unitarias son las que se definen antes de realizarse la implementación del código. Las pruebas de aceptación son creadas a partir de las historias de usuario, estas permiten determinar si cada historia de usuario ha sido implementada satisfactoriamente.

Pruebas Unitarias

Las pruebas unitarias o también llamada pruebas de caja blanca (White Box), estas pruebas también son llamadas pruebas modulares ya que nos permiten determinar si un módulo del programa está listo y correctamente terminado. Este tipo de pruebas debe ser realizado por personal el cual debe estar familiarizado en el uso de herramientas de depuración y pruebas, así mismo deben conocer el lenguaje de programación en el que se está desarrollando la aplicación. El objetivo fundamental de las pruebas unitarias es asegurar el correcto funcionamiento de las interfaces, o flujo de datos entre componentes. (40) Se le hicieron pruebas unitarias a las ocho funcionalidades más significativas del sistema, estas se pueden en los Anexos.

Pruebas de Aceptación

Se denominan pruebas de aceptación son consideradas pruebas de caja negra, tiene por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios finales, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta el paso siguiente es el pase a producción. (40) Son creadas a partir de la historias de usuario con el objetivo de inspeccionar que cada una de ellas se haya desarrollado correctamente. El cliente es el máximo responsable de verificar que los resultados de estas pruebas sean correctos. Una historia de usuario puede tener todas las pruebas de aceptación que sean necesarias para lograr determinar su completo funcionamiento y a su vez no puede ser considerada terminada hasta que no se le realicen sus pruebas pertinentes.

A continuación se presentan los casos de prueba de las HU más significativas, el resto se puede en los Anexos.

Tabla 18. Caso de Prueba de Aceptación HU1_P1

Caso de Prueba de Aceptación

Código: HU1_P1	HU: HU1
Nombre: Determinar la cantidad de permutaciones de n elementos.	
Descripción: Muestra el número de permutaciones de n elementos.	
Condiciones de Ejecución: El usuario puede visualizar el número de permutaciones que se derivan.	
Entradas\Pasos de Ejecución: El usuario introduce en introduce en la caja de texto el comando permutaciones, que puede ser de dos maneras: Si lo que quieres es calcular la cantidad de permutaciones de n elementos y ese n es un conjunto el comando seria: **permutaciones({1,2,3})**de lo contrario si es un número sería: **permutaciones(3).**	
Resultado Esperado: Se muestra un número que representa la cantidad de permutaciones.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 19. Caso de Prueba de Aceptación HU2

Caso de Prueba de Aceptación	
Código: HU2	HU: HU2
Nombre: Generar todas las permutaciones de un conjunto dado.	
Descripción: Muestra el listado de permutaciones que se genera de un conjunto dado.	
Condiciones de Ejecución: El usuario puede visualizar el listado de permutaciones que se derivan.	
Entradas\Pasos de Ejecución: El usuario introduce en introduce en la caja de texto el comando permutaciones, pasándole como parámetro un conjunto, dicho conjunto de debe ir entre llaves y los valores separados por comas, que puede ser de hasta 7 caracteres separados por comas, ej.: **permutaciones({1,ed,3e}).**	
Resultado Esperado: Se muestra un listado de conjuntos que representa las permutaciones que se derivan.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 20. Caso de Prueba de Aceptación HU4

Caso de Prueba de Aceptación	
Código: HU4	HU: HU4
Nombre: Determinar la cantidad de combinaciones posibles de r elementos de n.	
Descripción: Muestra un número que representa las cantidad de combinaciones posible de r elementos de n.	
Condiciones de Ejecución: El usuario puede visualizar el número de permutaciones que se derivan.	
Entradas\Pasos de Ejecución: El usuario introduce en introduce en la caja de texto el comando cantcombinaciones, que puede ser de dos maneras: en el parámetro n se le puede pasar un número o conjunto, mientras que r siempre será un número, los parámetros van separados por un punto y coma (;), y en caso de que n es un conjunto va entre llaves con los valores separados por una coma (,) ej.: **cantcombinaciones({1,ed,3e};2) o cantcombinaciones(3;2).**	
Resultado Esperado: Se muestra un número que representa las permutaciones que se derivan.	
Evaluación de la Prueba: Satisfactoria	

3.3 Conclusiones parciales del capítulo

En el capítulo se brindó la información pertinente a las fases de implementación y prueba. Se realizaron todas las tareas que permiten darle solución a sus correspondientes historias de usuario. También se muestran los resultados de las pruebas de aceptación realizadas al sistema con el objetivo de comprobar el correcto funcionamiento de las historias de usuario.

CONCLUSIONES GENERALES.

Después de la exhaustiva investigación realizada con el objetivo de facilitar una herramienta informática a los estudiantes de manera tal que constituya una herramienta de estudio de apoyo al proceso de enseñanza-aprendizaje de la asignatura Matemática Discreta, se plantean como conclusiones generales de este trabajo las siguientes:

- ✓ Se efectuó un estudio sobre las metodologías y herramientas existentes en el mundo para el desarrollo de aplicaciones matemáticas para obtener como resultado una correcta selección de la metodología y herramienta a utilizar, teniendo en cuenta las características del software a desarrollar.
- ✓ Se seleccionó XP como metodología, como lenguaje de programación Java, el IDE NetBeans herramienta de modelado Visual Paradigm, para el desarrollo de la aplicación.
- ✓ Se realizaron las fases de planificación, diseño, desarrollo y pruebas que propone la metodología XP, las que propiciaron el correcto desarrollo de la aplicación.
- ✓ Con la realización de esta herramienta se considera que los estudiantes de primer año que cursan la asignatura Matemática Discreta, así como todos los interesados, contarán con una herramienta que servirá de apoyo al proceso de enseñanza-aprendizaje, permitiendo fomentar, ejercitar y consolidar los conocimientos acerca de la Combinatoria y Teoría de Números.

RECOMENDACIONES.

Después de darle cumplimiento a los objetivos propuestos y de implementar el sistema, se recomienda:

- ✓ Confeccionar la ayuda y manual de usuario para facilitarle el trabajo a las personas que un futuro trabajen con el software.
- ✓ Validar el sistema y que este sea puesto en funcionamiento lo antes posible por parte de los interesados.
- ✓ La realización de nuevas versiones de la aplicación para ampliar las funcionalidades de la solución propuesta.
- ✓ Aplicar la solución propuesta al resto de la Universidad de las Ciencias informáticas.
- ✓ Tener en cuenta este trabajo para proyectos futuros, siempre y cuando se cumpla con las normas de confidencialidad establecidas

BIBLIOGRAFÍA.

1. **Bautista, Juan.** El Blog de Juan Bautista. [Online] noviembre 20, 2007. [Citado: noviembre 5, 2012.] http://comunidadesvirtuales.oblog.com/importancia-tic-proceso-ensennanza-aprendizaje. 40185.

2. **Fernandez, Inmaculada.** LAS TICS EN EL AMBITO EDUCATIVO. [Online] [Cited: diciembre 5, 2012.] http://www.eduinnova.es/abril2010/tic_educativo.pdf.

3. **Samuel.** [Online] marzo 19, 2012. [Cited: noviembre 9, 2012.] http://mat115matematicadiscreta.blogspot.com/2012/03/matematica-discreta.html..

4. **Ruiz Godoy, Juan Manuel, Ramirez Hernandez, Yanelis and Gonzalez del Sol, Sonia.***Propuesta de una estrategia para el proceso de enseñanza-aprendizaje con el uso de un asistente matematico. 2012. Vol. 5.*

5. ***Derive.*** [Online] [Cited: dieciembre 8, 2012.] http://www.chartwellyorke.com/derive.html..

6. ***Taringa.*** [Online] [Cited: diciembre 8, 2012.] http://x-taringa.net/848-maple-13-full-el-mejor-software-de-matematica.html..

7. **Garcia de Jalon, Javier, Rodriguez, Jose Ignacio and Vidal, Jesus.***Aprenda Matlab 7.0.* 2005. s.n..

8. ***Mathematica.*** [Online] [Cited: diciembre 8, 2012.] http://www.wolfram.com/mathematica/.

9. ***IDENTY.*** [Online] [Cited: diciembre 8, 2012.] http://www.identi.li/index.php?topic=64910..

10. **Lo discreto y lo continuo**. [Online] [Cited: diciembre 8, 2012.] http://www.sangakoo.com/blog/lo-discreto-y-lo-continuo/.

11. **Munoz Izquierdo, Roberto, et al.** Matematica Discreta. [Online] [Cited: febrero 5, 2012.] HTTP://WWW.ESCET.URJC.ES/~MATEMATI/MD_ITI/MD_ITI.HTML..

12. **Garrido, Luciano.**[Online] [Cited: febrero 5, 2013.] http://ingenieria.lm.uasnet.mx/sitio/servicios/pagper/maestros/manuel/Matematicas%20Discretas/apuntes%20de%20matematicas%20discretas%282%29.pdf..

13. **Scribd**. *Teoria de la Combinatoria.* [Online] [Cited: diciembre 10, 2012.] http://es.scribd.com/doc/92098284/Combinatoria-Ofe-1..

14. **Arcia Carrazana, Maikel.***Matematica Discreta UCI.* 2005.

15. **Rosen, Kenneth H.***Discrete Mathematics and Its Aplications. Seventh. 011 : 06, 2011. 978-0-07-338309-5.*

16. **Santos, David A.***Teoría elemental de los números para olimpíadas matemáticas*. 2005.

17**. Cobos Gavala, Fco. Javier.***Introduccion a la Matematica Discreta* . 2003 : s.n.

18. **Sanchez, M.** Metodologias De Desarrollo De Software. [Online] [Cited: enero 7, 2013.] http://www.willydev.net/descargas/cualmetodologia.pdf.

19. **Rational Sofware Corporation.***El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2001.

20. **Mendoza Sanchez, Maria.** Informatizate.net. [Online] 2004. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html..

21. **Penades, Maria del Carmen, Letelier, Patricio and Canos, Jose H.***Metodologias agiles para el desarrollo de software: eXtreme Programing(XP).* Valencia : s.n.

22. XPrograming.com. [Online] http://www.xprogramming.com/software.htm.

23. **Aguilar Sierra, Alejandro.** seguridad.unam.mx. *seguridad.unam.mx.* [Online] octubre 2002. http://www.seguridad.unam.mx/eventos/datos/ev11/semi18/mat.7.pon19.semi18.pdf...

24. **Wilson, Leslie B.***Comparative Programing Language.* Second Edition. s.l. : Addison-Wesley, 1993.

25. **Netbeans.** [Online] [Cited: enero 7, 2013.] http://netbeans.org/community/releases/68..

26. **Booch, Grady, Jacobson, Ivar and Rumbaugh, Jim.***El Lenguaje Unificado de Modelado.* s.l. : Addison-Wesley, 1999. 84-7829-028-1.

27. **Visual Paradigm.** [Online] 2009. http://www.visual-paradigm.com..

28. **Jacobson, Ivar, Booch, Grady and Rambaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison-Wesley, 2000.

29**. Fowler, Scott.***UML gota a gota.* s.l. : PRENTICE HALL, 1997.

30. **Fases de la Programacion Extrema**. [Online] [Cited: abril 5, 2013.] http://programacionextrema.tripod.com/fases.html.

31. **IEEE SA-1471-2000.** IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. [Online] http://standards.ieee.org/findstds/standard/1471-2000.html..

32**. Garlan, David and Shaw, Mary.***An Introduc-tion to Software Architecture.* s.l. : School of Computer Science Carnegie Mellon Univer-sity Pittsburgh, 1994.

33. **Kazman, R., Bass, L. and Clements, P.***Software Architecture in Practice.* 2003.

34. **De la Torre Llorente, Cesar, et al.***Guia de Arquitectura N-Capas orientada con .Net.* 4.0. s.l. : Krasis Consulting, 2010.

35. **Larman, C.***UML y Patrones. Introduccion al analisis y diseño orientado a objetos.* La Habana : Felix Varela, 2004.

36. **Hernandez, Luis.**Software Guisho. [Online] [Cited: 5 4, 2013.] http://software.guisho.com/singleton-pattern-patrones-de-diseño...

37. **Pressman, Roger S.***Ingenieria del Software. Un enfoque practico.* Sexta edicion.

38. **Drake, Jose M and Lopez, Patricia.** Verificacion y Validacion. [Online] 2009. http://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion-2011.pdf..

39. **Blanco Llano, Javier and Rodriguez Hernandez, Aida.***REVISIÓN, VERIFICACIÓN Y VALIDACIÓN EN UN PROCESO DE DESARROLLO.* 2011. ISSN 1815-5936.

40. **Re B., Alexander.** [Online] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.