Universidad de las Ciencias Informáticas

FACULTAD 3

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

SIMUCI: HERRAMIENTA DE APOYO A LA ASIGNATURA SIMULACIÓN

Autor: JOENIS RODRÍGUEZ FERNÁNDEZ.

Tutores: Ing. Ariam Jorge Pedrosa González.

MSc. Yaniesis Lorenzo Costa.

La Habana, junio 2016

“Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.”

Albert Einstein

Declaro que soy el único autor del trabajo titulado:

"SIMUCI: HERRAMIENTA DE APOYO A LA ASIGNATURA SIMULACIÓN".

Y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente en el mes de _____ del año _____.

_____

Joenis Rodríguez Fernández

_____ _____

Ing. Ariam Jorge Pedrosa González. MSc. Yaniesis Lorenzo Costa.

"Con todo mi cariño y mi amor para las personas que hicieron todo en la vida para que yo pudiera lograr mis sueños, por motivarme y quererme, a ustedes por siempre mi corazón y mi agradecimiento"

Mamá y Papá

Este trabajo de diploma representa en mi vida el acontecimiento que marca un antes y un después, la experiencia universitaria, camino recorrido lleno de ganancias y pérdidas; la entrega a un futuro que me invita a soñar y a luchar por mis sueños.

A mis padres, porque creyeron en mí y porque me sacaron adelante, dándome ejemplos dignos de superación y entrega, porque gracias a ustedes, hoy puedo ver alcanzada mi meta, ya que siempre estuvieron impulsándome en los momentos más difíciles de mi carrera, y porque el orgullo que sienten por mí, fue lo que me hizo ir hasta el final. Va por ustedes, por lo que valen, porque admiro su fortaleza y por lo que han hecho de mí.

RESUMEN

Desde su aparición y con el propósito de elevar continuamente la calidad de la educación en todos los sectores de la sociedad, las Tecnologías de la Información y las Comunicaciones se han integrado en la búsqueda de alternativas para realizar los tradicionales procesos académicos, suscitando a la introducción de sistemas educativos dentro del proceso de enseñanza aprendizaje y convirtiéndolo en un proceso más amplio, personalizado y flexible. El presente trabajo propone una herramienta de apoyo a la asignatura Simulación en la Universidad de las Ciencias Informáticas, la cual implementa un conjunto de los métodos que se imparten en dicha asignatura y que pueden ser ejecutados a decisión del usuario y en correspondencia a sus necesidades. Entre los principales aportes de esta herramienta para la asignatura Simulación con respecto a los métodos implementados, destacan: el bajo consumo de tiempo en la realización de cálculos auxiliares, la presentación de resultados parciales paso a paso durante su ejecución, y la facilidad de realizar, a partir de un mismo juego de datos, varias ejecuciones de métodos similares y comparar los resultados obtenidos. El desarrollo de SIMUCI se realizó bajo la concepción del marco de trabajo que propone el departamento de Ciencias Básicas de la Facultad 3 para la construcción de herramientas que apoyen sus asignaturas. Asimismo, se utilizó la metodología de software eXtreme Programming para generar los artefactos correspondientes a cada una de sus fases y guiar su proceso de desarrollo. La herramienta fue probada por especialistas en calidad de software y aceptada por los profesores del departamento de Ciencias Básicas de la Facultad 3.

Palabras claves: ciencias básicas, proceso de enseñanza aprendizaje, software educativo, simulación

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO I: EL SOFTWARE EDUCATIVO COMO MEDIO DE ENSEÑANZA 8

1.1. INTRODUCCIÓN 8
1.2. CONCEPTUALIZACIÓN, IMPORTANCIA Y FUNCIONES DEL SOFTWARE EDUCATIVO 8
1.3. CARACTERÍSTICAS DE LA ASIGNATURA SIMULACIÓN EN LA UCI 13
1.4. DESCRIPCIÓN DEL MARCO DE TRABAJO PROPUESTO POR EL DEPARTAMENTO DE CIENCIAS BÁSICAS PARA LA CONSTRUCCIÓN DE HERRAMIENTAS QUE APOYEN A SUS ASIGNATURAS 17
1.5. METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE 18
1.6. CONCLUSIONES DEL CAPÍTULO 22

CAPÍTULO II: PROPUESTA DE SOLUCIÓN 25

2.1. INTRODUCCIÓN 25
2.2. DESCRIPCIÓN DEL NEGOCIO 25
2.3. PLANEACIÓN 26
2.3.1. Historias de usuario *26*
2.3.2. Requisitos de software *28*
2.3.3. Línea base tecnológica *30*
2.3.4. Estimación del esfuerzo por historia de usuario. Planes de iteraciones, duración de iteraciones y de entrega *34*
2.4. DISEÑO 36
2.4.1. Tarjetas CRC *36*
2.4.2. Patrón arquitectónico *38*
2.4.3. Patrones de diseño *40*
2.4.4. Prototipos de interfaz de usuario *41*
2.5. CONCLUSIONES DEL CAPÍTULO 43

CAPÍTULO III: CODIFICACIÓN Y VALIDACIÓN DE LA SOLUCIÓN 45

3.1. INTRODUCCIÓN 45
3.2. CODIFICACIÓN 45
3.2.1. Estándares de codificación *45*
3.2.2. Codificación de pruebas *47*
3.3. PRUEBAS DE SOFTWARE 48
3.3.1. Pruebas unitarias *49*
3.3.2. Pruebas de aceptación *49*
3.3.4. Validación de los resultados obtenidos *53*
3.4. CONCLUSIONES DEL CAPÍTULO 55

CONCLUSIONES FINALES 57

RECOMENDACIONES 58

GLOSARIO DE TÉRMINOS 59

REFERENCIAS BIBLIOGRÁFICAS 60

ANEXOS 63

ÍNDICE DE TABLAS

TABLA 1: RANKING DE AGILIDAD 21
TABLA 2: DESCRIPCIÓN GENERAL DE UNA HU 27
TABLA 3: DESCRIPCIÓN DE LA HU1 (ELABORACIÓN PROPIA) 27
TABLA 4: ESTIMACIÓN DEL ESFUERZO POR HU. PLANES DE ITERACIONES, DURACIÓN DE ITERACIONES Y DE ENTREGA (ELABORACIÓN PROPIA) 35
TABLA 5: DESCRIPCIÓN DE LA TARJETA CRC CORRESPONDIENTE A LA CLASE VIEWS.PY (ELABORACIÓN PROPIA) 37
TABLA 6: ESTÁNDARES DE CODIFICACIÓN UTILIZADOS EN SIMUCI (ELABORACIÓN PROPIA) 46
TABLA 7: DISEÑO DE CASO DE PRUEBA DE ACEPTACIÓN PARA LA HU9 (ELABORACIÓN PROPIA) 52
TABLA 8: JUEGO DE DATOS PARA EL CASO DE PRUEBA DE ACEPTACIÓN DE LA HU9 (ELABORACIÓN PROPIA) 52
TABLA 9: CUADRO LÓGICO DE IADOV (ELABORACIÓN PROPIA) 53
TABLA 10: RELACIÓN ENTRE LA SATISFACCIÓN INDIVIDUAL Y LA ESCALA DE SATISFACCIÓN (ELABORACIÓN PROPIA) 54

ÍNDICE DE FIGURAS

FIGURA 1: ESQUEMA FUNCIONAL DE SIMUCI (ELABORACIÓN PROPIA) 26
FIGURA 2: EJEMPLO DEL PATRÓN MPV EN SIMUCI (ELABORACIÓN PROPIA) 40
FIGURA 3: PROTOTIPO DE INTERFAZ: PANTALLA PRINCIPAL CON MENÚ DESPLEGADO 42
FIGURA 4: PROTOTIPO DE INTERFAZ: TEST DE KOLMOGOROV-SMIRNOV 42
FIGURA 5: CASO DE PRUEBA CORRESPONDIENTE A LA HU9 (ELABORACIÓN PROPIA) 48
FIGURA 6: PASE SATISFACTORIO DEL CASO DE PRUEBA DEFINIDO PARA LA HU9 (ELABORACIÓN PROPIA) 49
FIGURA 7: RESULTADOS DE LAS PRUEBAS DE ACEPTACIÓN (ELABORACIÓN PROPIA) 50
FIGURA 8: TRATAMIENTO DE LAS NC EN LAS ITERACIONES (ELABORACIÓN PROPIA) 51
FIGURA 9: RESUMEN DE LOS RESULTADOS INDIVIDUALES OBTENIDOS (ELABORACIÓN PROPIA) 54

INTRODUCCIÓN

La educación constituye uno de los pilares más importantes de la sociedad. Con el transcurso de los años se ha desarrollado y perfeccionado con el objetivo de crear instituciones educativas de excelencia. Las universidades del siglo XXI necesitan ser ante todo competitivas, ambiciosas, vinculadas con sus realidades sociales y económicas, activas y flexibles. Además, deben centrarse en primer lugar en el aprendizaje a gran escala y en el aprovechamiento de todos sus recursos humanos y materiales. Asimismo anticiparse a los cambios sociales y económicos, establecer métodos educativos innovadores, trabajar por una máxima gestión de la información y tener como meta introducirse de forma comprometida en la actual y desarrollada sociedad del conocimiento; resultado imprescindible de la incorporación de las nuevas Tecnologías de la Información y las Comunicaciones (TIC) (LAVIÑA ORUETA Y MENGUAL PAVÓN 2010).

Las TIC son un conjunto de medios o herramientas tecnológicas de la informática y la comunicación que constituyen un elemento clave en el desarrollo de la educación, cambiando considerablemente los métodos tradicionales de enseñanza. La facilidad de crear, procesar y difundir información ha superado todas las barreras que limitan la adquisición del conocimiento, contribuyendo al desarrollo de habilidades y destrezas comunicativas entre docentes y estudiantes. La implementación de la tecnología en la educación no persigue sustituir al maestro, sino ayudarle para que el estudiante se nutra de más elementos que enriquezcan y perfeccionen el Proceso de Enseñanza Aprendizaje (PEA).

Cuba, a pesar de las limitaciones económicas que posee, conserva una fuerte voluntad política de elevar el nivel y la calidad de la educación superior. Una muestra de ello es la creación de la Universidad de las Ciencias Informáticas (UCI). La UCI tiene como misión formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación y servir de soporte a la industria cubana de la informática (UCI 2012).

Entre las alternativas que asume la UCI para favorecer el proceso de enseñanza-aprendizaje (PEA) adquiere una connotación especial, el logro de un enfoque interdisciplinario en la docencia. La interdisciplinariedad puede verse como una estrategia pedagógica que implica la interacción de varias disciplinas, entendida como el diálogo y la colaboración de estas

para lograr la meta de un nuevo conocimiento (VAN DER LINDE 2014). Un ejemplo de campo interdisciplinario dentro de la UCI es el relacionado a las Ciencias Básicas, donde asignaturas como Matemática Numérica, Física, Álgebra, Probabilidad y Estadísticas, Investigación de Operaciones y Simulación juegan un rol fundamental en la formación del futuro profesional en ciencias informáticas, sin embargo, según los profesores que imparten estas asignaturas en la Facultad 3 de la UCI, al ser no troncales, es decir, que no pertenecen a la especialidad, tienden a ser menos asimiladas por los estudiantes.

En el departamento de Ciencias Básicas de la Facultad 3 de la UCI, se está apostando por el diseño e implementación de un conjunto de soluciones bajo la concepción de un mismo marco de trabajo como apoyo a las asignaturas asociadas a dicho departamento. En un principio se comenzó con MATHNUM®, potente instrumento matemático que facilita la representación numérica y geométrica de las soluciones aproximadas según los métodos estudiados en la asignatura Matemática IV (SARIOL FERNÁNDEZ 2015), siendo hasta ahora la única herramienta implementada. La idea de incorporar herramientas de apoyo a dichas asignaturas viene dada principalmente por la poca motivación que sienten los estudiantes de la carrera por el estudio de las Ciencias Básicas. Además, los profesores del área consideran que la introducción de herramientas al PEA proveerá nuevos mecanismos para que los estudiantes se nutran de los conocimientos necesarios y básicos de cada disciplina.

Entre las asignaturas que aún no cuentan con una herramienta de apoyo al PEA se encuentra Simulación. En su lugar y con el objetivo de lograr una mayor compresión de sus principales conceptos, son utilizados algunos asistentes matemáticos como MatLab, Mathematica y R. Sin embargo, estos sistemas no están diseñados para fines docentes, van directo a obtener el resultado final pasando por alto incluso los resultados intermedios más importantes, que ayudan en el análisis e interpretación del contenido, además demandan de una gran cantidad de recursos computacionales para su ejecución. Son herramientas propietarias, y en algunos casos como MatLab, están expresamente bloqueados para Cuba. El trabajo para los estudiantes de ingeniería con estas herramientas, especialmente en la asignatura Simulación, continúa siendo un problema en cuanto a:

- La alta complejidad que presentan, demandando la necesidad de un conocimiento avanzado de los métodos empleados y el dominio de su sintaxis particular.

- La no presentación de algunos resultados parciales e iteraciones de métodos, limitan su aplicación a la comprobación de los resultados obtenidos.

Atendiendo a la problemática descrita anteriormente se define como **problema a resolver**: ¿Cómo desarrollar una herramienta informática bajo la concepción del marco de trabajo propuesto por el departamento de Ciencias Básicas de la Facultad 3, que contribuya a la visualización numérica, paso a paso, de los métodos que se imparten en la asignatura Simulación?

En correspondencia con el problema, se precisa como **objeto de estudio**: el proceso de desarrollo de software; siendo el **campo de acción:** el proceso de desarrollo de software educativo.

Para dar solución a la situación planteada se establece como **objetivo general** de la investigación: desarrollar una herramienta informática bajo la concepción del marco de trabajo propuesto por el departamento de Ciencias Básicas de la Facultad 3, que contribuya a la visualización numérica, paso a paso, de los métodos que se imparten en la asignatura Simulación; derivándose en los siguientes **objetivos específicos**:

1. Establecer el marco teórico de la investigación mediante el estudio de los referentes teóricos del proceso de desarrollo de software educativo, así como las características de la asignatura Simulación en la UCI y el marco de trabajo propuesto por el departamento de Ciencias Básicas de la Facultad 3.
2. Generar los artefactos necesarios para la construcción del modelo de una herramienta informática que contribuya a la visualización numérica, paso a paso, de los métodos que se imparten en la asignatura Simulación, mediante la metodología de desarrollo de software seleccionada.
3. Implementar la herramienta informática utilizando la línea base tecnológica que propone el marco de trabajo del departamento de Ciencias Básicas de la Facultad 3 para el desarrollo de soluciones que apoyen a las asignaturas de dicho departamento y a partir de los artefactos generados por la metodología seleccionada.
4. Valorar la efectividad de la solución propuesta mediante pruebas de software y la aplicación de la técnica Iadov.

Para el cumplimiento de los objetivos se definen las siguientes **tareas de investigación**:

1. Análisis de los referentes teóricos del proceso de desarrollo de software en el ámbito educativo para la asignatura Simulación.

2. Análisis de las características de la asignatura Simulación en la UCI.
3. Análisis del marco de trabajo propuesto por el departamento de Ciencias Básicas de la Facultad 3 para el desarrollo de herramientas que apoyen a sus asignaturas.
4. Selección de la metodología a utilizar para el desarrollo de una herramienta informática que contribuya a la visualización numérica, paso a paso, de los métodos que se imparten en la asignatura Simulación.
5. Generación de los artefactos necesarios para la construcción del modelo de una herramienta informática que contribuya a la visualización numérica, paso a paso, de los métodos que se imparten en la asignatura Simulación.
6. Implementación de la herramienta informática a partir de la línea base tecnológica propuesta por el marco de trabajo del departamento de Ciencias Básicas de la Facultad 3.
7. Ejecución de pruebas unitarias al código fuente obtenido de la herramienta implementada.
8. Valoración de la efectividad de la herramienta implementada mediante pruebas de aceptación con el cliente y la aplicación de la técnica Iadov.

Durante el desarrollo de la presente investigación se utilizaron los siguientes **métodos científicos,** luego de estudiar las consideraciones de (HERNÁNDEZ SAMPIERI *et al.* 2010).

Métodos teóricos:

- **Analítico-Sintético:** divide mentalmente el estudio en diferentes áreas según los objetivos: analiza, valora y evalúa la información. Luego permite unir las partes previamente analizadas descubriendo las características y relaciones entre ellas. Usado generalmente durante el estudio de los sistemas educativos, el marco de trabajo propuesto por el departamento de Ciencias Básicas de la Facultad 3 y las características de la asignatura Simulación en la UCI.
- **Histórico-lógico:** establece la necesaria correspondencia entre los elementos de los métodos lógico e histórico, proyectando el análisis de la evolución histórica de los fenómenos, con la proyección lógica de su comportamiento futuro. Utilizado para el estudio de la evolución e impacto actual de los sistemas educativos y las TIC.
- **Modelación:** método que opera en forma práctica o teórica con un objeto, no en su forma directa, sino utilizando cierto sistema intermedio, auxiliar, natural o artificial.

Mediante la modelación se crearon abstracciones del sistema propuesto como las tarjetas CRC.

Métodos empíricos:

- **Encuesta:** práctica que permite la adquisición de información de interés sociológico, mediante un cuestionario previamente elaborado, a través del cual se puede conocer la opinión o valoración del sujeto seleccionado en una muestra sobre un asunto dado. Utilizada para conocer la percepción de especialistas y estudiantes sobre el sistema que se desarrolla y su impacto para la universidad.
- **Entrevista:** implica la recopilación de información mediante una conversación profesional. Los resultados a lograr dependen en gran medida del nivel de comunicación entre el investigador y los participantes en la misma. Fueron puestas en práctica mayoritariamente con los profesores del departamento, para conocer la forma en que se realizan los diferentes métodos que se imparten en la asignatura Simulación.

Para una mejor comprensión, el presente documento se estructura en: introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

En el **CAPÍTULO I,** denominado **"El SOFTWARE EDUCATIVO COMO MEDIO DE ENSEÑANZA",** se investiga el impacto que ha tenido el desarrollo del software educativo como medio de apoyo al PEA. Se incluye un epígrafe donde se describen las particularidades que tiene la asignatura Simulación en la UCI. Además se realiza un análisis del marco de trabajo que propone el departamento de Ciencias Básicas de la Facultad 3 de la UCI para la construcción de herramientas que apoyen sus asignaturas. A su vez, desde los principios de los enfoques ágiles de la Ingeniería del Software, se propone el uso de una metodología para regir el diseño y desarrollo de la aplicación. En un último momento se presentan las conclusiones del capítulo.

En el **CAPÍTULO II,** referido como **"PROPUESTA DE SOLUCIÓN",** se abarcan los principales artefactos que describen el comportamiento de la aplicación en cuanto a funcionalidades y demás características que presenta según lo establecido en la metodología de software seleccionada en las fases de planeación y diseño. Durante este capítulo se construyeron las historias de usuario descritas por los clientes, se derivaron los requisitos de la aplicación, y se caracterizaron las herramientas y tecnologías capaces de

cumplir dichos requisitos. En relación a las historias de usuario definidas, se elaboraron en conjunto los planes de iteraciones, duración de iteraciones y de entregas. Además son descritas las tarjetas CRC que se utilizaron y los patrones Modelo – Plantilla – Vista, controlador, bajo acoplamiento y alta cohesión. En un último momento se presentan las conclusiones del capítulo.

En el **CAPÍTULO III,** descrito como **"CODIFICACIÓN Y VALIDACIÓN DE LA SOLUCIÓN",** se caracterizan los artefactos generados como parte de las fases de codificación y pruebas definidas en la metodología de software seleccionada. Se establecieron los estándares de programación para la propuesta de solución y se codificaron las pruebas unitarias. Además, se muestran los resultados obtenidos luego de aplicadas las pruebas unitarias y de aceptación a la aplicación. Además se determina el grado de satisfacción de una muestra de estudiantes para con la herramienta implementada utilizando la técnica Iadov. En un último momento se presentan las conclusiones del capítulo.

CAPÍTULO I:

EL SOFTWARE EDUCATIVO COMO MEDIO DE ENSEÑANZA

CAPÍTULO I: EL SOFTWARE EDUCATIVO COMO MEDIO DE ENSEÑANZA

1.1. Introducción

El presente capítulo tiene como objetivo el análisis crítico sobre la pertinencia de incorporar un software educativo como medio de apoyo a la asignatura Simulación en la UCI. Para cumplir con este objetivo se realiza un estudio del estado actual de las características del software educativo, así como sus ventajas, y funciones principales que puede adoptar en dependencia de la manera en que el profesor organice su utilización dentro del PEA. De forma similar, se presenta un análisis sobre el marco de trabajo que propone el departamento de Ciencias Básicas de la Facultad 3 de la UCI para el desarrollo de herramientas que apoyen a las asignaturas del departamento, y se describen las características de la asignatura Simulación en la UCI, brindando las consideraciones referentes a las mejoras que podrían representar la introducción de una herramienta informática en sus clases. Además, luego de un análisis de diferentes metodologías para el desarrollo de software se realiza una propuesta para regir la investigación. En un último momento se presentan las conclusiones del capítulo.

1.2. Conceptualización, importancia y funciones del software educativo

El desarrollo acelerado de la ciencia y la tecnología en el campo de la Informática y las Comunicaciones ha provocado que en la actualidad se implementen aplicaciones informáticas en todos los sectores de la sociedad. Específicamente, en el ámbito académico, se ha impulsado el uso y desarrollo del software como medio para potenciar la calidad del PEA de las futuras generaciones. El software educativo posibilita la transmisión de conocimientos de una forma más amena, integradora, diferenciada, reguladora y activa que el resto de los medios de enseñanza (LABRADA 2011).

Un medio de enseñanza tiene como misión fundamental facilitar o apoyar el aprendizaje de los alumnos. Estos sistemas se utilizan mayoritariamente como refuerzo de la acción del profesor en clase y en situaciones con la presencia del profesor, facilitando y mejorando la comunicación con los alumnos. En otros casos, también seleccionados y controlados por el profesor, se pueden mostrar autosuficientes para la explicación de un contenido (RAMOS 2004).

De acuerdo con (SÁNCHEZ, TOCTAQUIZA Y MARCELO 2013), diversos autores de las ciencias pedagógicas han abordado desde sus investigaciones el concepto de software educativo:

- (SÁNCHEZ, JAIME *et al.* 1999) define el concepto genérico de software educativo como cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar.
- (RODRÍGUEZ LAMAS *et al.* 2000) lo precisa como una aplicación informática que soportada sobre una bien definida estrategia pedagógica, apoya directamente el PEA constituyendo un efectivo instrumento para el desarrollo educacional del hombre.
- (LABAÑINO RIZO 2005) lo detalla como una aplicación informática concebida especialmente como medio integrado al PEA.
- (LABRADA 2011) lo enuncia como un medio didáctico digital autónomo, elaborado por un equipo multidisciplinario, encaminado al desarrollo de la personalidad de los educandos desde el punto de vista afectivo y cognitivo a partir de la integración de recursos y en correspondencia con los objetivos del currículo de la enseñanza y los destinatarios a que está dirigido.

La mayoría de los autores coinciden en las definiciones aportadas, el carácter instrumental del software educativo, a la vez que lo refieren como cualquier programa de computador creado específicamente para apoyar y ser utilizado en el PEA. La importancia de su desarrollo según (SÁNCHEZ, TOCTAQUIZA Y MARCELO 2013) radica en las siguientes afirmaciones:

- Son altamente interactivos, a partir del empleo de recursos, como videos, sonidos, fotografías, hipertextos, diccionarios especializados, ejercicios y juegos instructivos que apoyan las funciones de evaluación y diagnóstico, permitiendo contestar inmediatamente las acciones de los estudiantes a través de un diálogo mutuo entre estos y el ordenador.
- Flexibilidad en su uso, tanto físico como de horario. El estudiante es capaz de decidir cuándo y cómo desea usarlo.
- Individualizan el trabajo; se adaptan al ritmo de trabajo de cada estudiante.
- Los estudiantes aprenden con rapidez y facilidad.
- Estimulan la construcción de conocimientos, nuevas formas de pensar, investigar y aprender haciendo.

- Desarrollan los procesos lógicos del pensamiento, la imaginación, la creatividad y la memoria.
- Facilitan las representaciones de procesos no perceptibles por el ojo humano en tiempo y espacio de forma animada.
- Simulan procesos complejos y optimizan el tiempo del que se dispone para impartir gran cantidad de conocimientos de forma amena y regulada por el usuario.
- Inciden en el desarrollo de habilidades psicomotrices y permiten retroalimentar al estudiante evaluando lo aprendido.

El desarrollo del software educativo tiene como base el poder desarrollar aplicaciones que soporten efectivamente el PEA. Es así, como el uso de las TIC abre nuevas posibilidades de innovación y realización de diferentes modelos pedagógicos que junto con la intrepidez, curiosidad y motivación del maestro para con los estudiantes, tienda a mejorar y cambiar de una forma positiva el proceso educativo (COLLAGUAZO Y ALEJANDRO 2016). La herramienta que pretende ser desarrollada como medio de enseñanza para la asignatura Simulación en la UCI, denominada SIMUCI:

- Será concebida con un propósito específico: apoyar la labor del departamento de Ciencias Básicas de la Facultad 3 en el proceso de aprendizaje de los estudiantes, por lo cual resulta necesario tener presente las características mencionadas anteriormente y definir cuáles incorporará.
- Permitirá ser accedida desde cualquier dispositivo electrónico con un navegador.
- Contestará inmediatamente las acciones de los estudiantes a partir del empleo de diferentes recursos.
- Optimizará el tiempo que dispone el profesor para impartir conocimientos.
- Mostrará la ejecución, paso a paso, de los métodos que implemente y de forma similar a como se estudian en la asignatura.

En consecuencia, con los análisis realizados hasta el momento y para cumplir con lo establecido en el primer punto de la lista previa, además de los elementos anteriores, SIMUCI tendrá como características:

- Flexibilidad en su uso: para ello deberá estar disponible en la web y poder ser accedida en cualquier momento, además deberá presentar los contenidos de forma intuitiva y organizada para el estudiante, siguiendo un orden lógico similar a como se imparten en las clases.

- Individualizará el trabajo adaptándose al ritmo de cada estudiante: la propia característica anterior promueve el hecho de que cada alumno pueda acceder indistintamente a la herramienta, utilizarla acorde a sus necesidades y ejecutando los métodos las veces que considere necesarias hasta lograr asimilar el contenido.
- Facilitará la presentación de procesos complejos: para lograrlo SIMUCI brindará las soluciones de todos sus métodos de forma “paso a paso”, permitiendo que los alumnos comprendan el proceso lógico que sigue cada función y asimilando la conversión de parámetros de entradas a valores de salida.
- Simulará procesos complejos optimizando el tiempo dedicado a los mismos.

Los programas didácticos, como también se les conoce, cuando se aplican a la realidad educativa, realizan funciones básicas y además en algunos casos, en dependencia de la manera en que el profesor organice su utilización y aplique la misma en el salón de clases, podrá desempeñar diferentes funciones. Sin embargo, como ocurre con otros sistemas educativos, no se puede afirmar que el software educativo por sí mismo sea bueno o malo, todo dependerá del uso que de él se haga y de la manera de cómo se utilice en cada situación concreta. En última instancia, su funcionalidad, ventajas e inconvenientes que pueda reportar su uso, serán el resultado de las características del material, de su adecuación al contexto educativo al que se aplica y de la manera en que el profesor organice su utilización (GARCÍA 2011). Por tal motivo es importante analizar las características de la asignatura Simulación en la UCI para que SIMUCI no esté desvinculada de su realidad. Este análisis se realizará un poco más adelante.

Resulta importante también estudiar las funciones que puede lograr un software educativo en los alumnos, según se describe en (SÁNCHEZ, TOCTAQUIZA Y MARCELO 2013). Estas funciones son:

- Función informativa: Presentan contenidos que proporcionan información estructurada de la realidad a los estudiantes.
- Función instructiva: Promueven determinadas actuaciones en los estudiantes, las cuales están encaminadas a facilitar el logro de los objetivos educativos específicos.
- Función motivadora: Los estudiantes se sienten atraídos e interesados por todo el software educativo. Los programas suelen incluir elementos capaces de captar la atención de los alumnos, mantener su interés y cuando sea necesario, enfocarlo hacia los aspectos más importantes de las actividades.

- Función evaluadora: La interactividad propia de estos materiales, que les permite responder inmediatamente a las respuestas y acciones de los estudiantes, los hacen especialmente adecuados para evaluar el trabajo que se realice con ellos. Esta evaluación puede ser de dos tipos:
    a. Implícita, cuando el estudiante detecta sus errores, se evalúa, a partir de las respuestas que le da el ordenador.
    b. Explícita, cuando el programa presenta informes valorando la actuación del alumno. Este tipo de evaluación sólo lo realizan los programas que disponen de módulos específicos de evaluación.
- Función investigadora: Ofrecen a los estudiantes interesantes entornos donde investigar y buscar determinadas informaciones.
- Función innovadora: Aunque no siempre los planteamientos pedagógicos que proponen resulten innovadores, el software educativo se puede considerar material didáctico con esta función, pues utiliza tecnología recientemente incorporada a los centros educativos y, en general, suele permitir muy diversas formas de uso. Esta versatilidad abre amplias posibilidades de experimentación didáctica e innovación educativa en el aula.
- Función lúdica: El software educativo tiene como finalidad reforzar de manera lúdica el conocimiento a través de juegos interactivos donde el estudiante adquiere las habilidades propuestas de una manera divertida.

De las funciones anteriores, la propuesta de solución a implementar, según las necesidades de los profesores de la asignatura Simulación en la Facultad 3 de la UCI, deberá cumplir, en primer lugar, con las siguientes:

- Función instructiva: el software deberá ser capaz de guiar al estudiante durante su interacción con la aplicación a través de una correcta estructuración de menús, con la finalidad de que el alumno adquiera los conocimientos necesarios y pueda utilizarlos para cumplir los objetivos de la asignatura.
- Función informativa: SIMUCI representará de forma estructurada y ordenada los contenidos que se imparten en la asignatura Simulación. Además, estos contenidos deben estar presentados similar a como los estudiantes lo reciben en las aulas.
- Función motivadora: la forma de organizar y representar el contenido deber ser amigable para los alumnos, la información y resultados de los métodos deben resaltar

por encima de lo demás, brindándole importancia a los elementos significativos en los cálculos y operaciones.

Luego de los análisis realizados hasta el momento y determinadas las características y principales funciones que debe cumplir la herramienta a desarrollar, respondiendo al *¿cómo?* realizará sus funciones, resulta de vital importancia analizar las peculiaridades de la asignatura Simulación en la UCI para, a partir de los contenidos que se estudian en la misma, conocer el *¿qué?* deberá implementar SIMUCI para ajustarse a las peculiaridades de la asignatura y responda funcionalmente a las necesidades de los clientes. El siguiente epígrafe estará dedicado a describir dichas características.

1.3. Características de la asignatura Simulación en la UCI

La asignatura Simulación comienza a impartirse en la UCI en el segundo semestre del curso escolar 2014-2015 para los estudiantes del cuarto año de la carrera. La asignatura posee un total de 48 horas clases divididas en dos temas:

- Tema 1: Introducción a la Simulación. Generación de variables aleatorias
- Tema 2: Simulación de eventos discretos

Simulación persigue como objetivos educativos:

1. Aplicar el rigor científico en la solución de los problemas que se abordan en la asignatura mediante la comprensión de la naturaleza y objetividad de los fenómenos y procesos de carácter aleatorio que estos representan, sobre la base de la aplicación de los conceptos, leyes y principios en que se sustentan, así como de los métodos y formas de trabajo que la componen.
2. Desarrollar la capacidad de razonamiento, el pensamiento lógico propio de la asignatura y el nivel de abstracción necesario mediante su participación activa en el PEA, con el análisis y solución de posibles situaciones prácticas bajo la dirección y guía del profesor.
3. Mostrar constancia en el estudio, apoyada e inducida mediante el diseño, desarrollo y control de un sistema de evaluación en la asignatura que permitan valorar, de forma sistemática, el grado alcanzado en el cumplimiento de los objetivos generales de estas, con el fin de estimular su actuación.

4. Desarrollar hábitos de trabajo independiente a partir del estudio de contenidos a través de la auto preparación respaldado por textos, literatura científica técnica y otros materiales orientados por el profesor.
5. Enfatizar en su trabajo en aspectos tales como eficiencia económica, ahorro, uso racional de la energía y de los recursos materiales y laborales, aumento de la productividad y cumplimiento de las normas del trabajo, así como la correcta preparación para todas las actividades y la entrega en tiempo y con calidad de informes y tareas, incluyendo en ello el uso correcto de nuestro idioma.

Y como objetivos instructivos:

1. Asimilar los conceptos básicos necesarios para la comprensión de los problemas de sistemas de servicios en particular y otros procesos discretos, en general, identificando los elementos que caracterizan a estos tipos de problemas.
2. Aplicar métodos matemáticos analíticos a la solución de este tipo de problemas cuando ello sea posible.
3. Resolver estos y otros problemas de decisión, a través del análisis de la simulación de las distintas alternativas planteadas con la ayuda de las computadoras.
4. Interpretar económicamente las soluciones obtenidas y evaluar con rigor estadístico las soluciones planteadas.
5. Identificar los elementos que componen el modelo de simulación de un proceso o sistema, y sus aplicaciones.
6. Aplicar las herramientas de la simulación de procesos y sistemas, básicamente el caso discreto, a problemas prácticos o hipotéticamente prácticos del entorno.

Una vez culminada la asignatura, los estudiantes deben ser capaces de:

1. Identificar los elementos que caracterizan a los sistemas discretos: sistemas de servicio y otros.
2. Identificar los elementos que componen un modelo de simulación discreta.
3. Aplicar métodos para generar números y variables aleatorias.
4. Aplicar la simulación en el planteamiento y solución de problemas discretos de dirección o toma de decisiones, aplicando el software existente cuando ello sea posible.

5. Analizar e interpretar económicamente, con ayuda de métodos estadísticos adecuados, la solución obtenida y seleccionar la mejor alternativa o proponer mejoras de solución.

Los profesores de la asignatura alegan que en la mayoría de los casos los estudiantes tienden a sentirse desmotivados por el exceso de cálculos auxiliares que deben realizar para obtener algunas soluciones, obviando muchas veces, el propio propósito de obtener un resultado para ser analizado o utilizado en otro momento e influyendo en el cumplimiento de los objetivos de la asignatura. Es por ello que el departamento de Ciencias Básicas de la Facultad 3 de la UCI, a raíz de este problema en todas sus asignaturas, apuesta por el desarrollo de herramientas informáticas que apoyen al PEA de las mismas, agilizándolo, haciéndolo más flexible y menos monótono para los estudiantes.

Si se analiza el contenido del Tema 1 de la asignatura Simulación se puede apreciar que se incluyen varios métodos donde debería resultar más importante el resultado del mismo y no el proceso para alcanzarlo. Sin embargo, no es suficiente el tiempo en clases para realizarlos y además incentivar el análisis de los resultados. Estos métodos son:

Para la generación de números aleatorios:

- Método congruencial aditivo
- Método congruencial multiplicativo
- Método congruencial mixto
- Generadores de Tausworthe

Para la generación de observaciones de variables aleatorias:

- Método de la transformada inversa
- Método de convolución
- Método de composición
- Método de la normal

Para la validación de sucesiones de números aleatorios:

- Test de Kolmogorov-Smirnov
- Test de Chi Cuadrado
- Test de las carreras por encima y por debajo de la media

- Test de autocorrelación
- Test de los huecos

Para el análisis de los resultados de la simulación

- Cálculo del intervalo de confianza para una variable aleatoria

Podría pensarse en la posibilidad de que los profesores de la asignatura y los estudiantes manipulen una herramienta computarizada que posea estos métodos y permita utilizarlos de manera rápida y eficiente. De existir los estudiantes podrán ver reflejado la ejecución de los mismos y podrán comparar los resultados que se obtienen cuando se aplican diferentes métodos a iguales datos de entrada. Además, podrían utilizarse de manera homóloga a como se les imparte a los estudiantes en las clases. De esta forma los alumnos no se sienten desvinculados, ni desconocen el algoritmo que se ejecuta por debajo de la interfaz de usuario.

Muy similar al Tema 1, ocurre en el Tema 2 de la asignatura: Simulación de eventos discretos. Como su propio nombre indica se debe realizar un proceso que, de no ser automatizado, no generará valores perceptibles de la realidad y apegados a situaciones concretas. De contarse con dicha herramienta que los profesores y estudiantes pudieran explotar para hacer simulaciones de sistemas a grandes escalas, sin la necesidad de desarrollar manualmente la inmensidad de cálculos matemáticos que esto genera, se podría dedicar mayor tiempo en clases al análisis de las simulaciones y la interpretación de los resultados. De esta forma se estaría cumpliendo el propósito de la asignatura en el año, que es en sí, que los estudiantes alcancen la capacidad de desarrollar un pensamiento lógico, con base en las matemáticas.

Por tales motivos, resulta imperante el uso de herramientas informáticas que agilicen el proceso de los cálculos en la asignatura Simulación que se imparte en la UCI y de esta forma mitiguen el consumo de tiempo en las clases dedicado a la realización de operaciones matemáticas. Este tiempo se puede aprovechar mucho más en el análisis de resultados, la interpretación de las salidas que brindan los sistemas y el apoyo a la toma de decisiones.

Una vez establecidas las particularidades que debe cumplir la herramienta que se desea desarrollar, luego de los estudios de las características del software educativo y la asignatura Simulación en la UCI, es necesario analizar el marco de trabajo que propone el departamento de Ciencias Básicas de la Facultad 3. Luego se podrán determinar las

herramientas y tecnologías que se utilizarán en la fase de diseño y codificación de SIMUCI. El siguiente epígrafe relaciona la concepción y justificación del establecimiento de dicho marco de trabajo.

1.4. Descripción del marco de trabajo propuesto por el departamento de Ciencias Básicas para la construcción de herramientas que apoyen a sus asignaturas

El universo de herramientas y tecnologías que persiguen como objetivo apoyar los procesos de análisis matemáticos, análisis estadísticos, solución de modelos matemáticos (lineales y no lineales), inferencia estadística y otros procedimientos matemáticos es inmenso. El número de soluciones y su heterogeneidad es tan diverso como las necesidades que las generan, principalmente provenientes del mundo académico y del mundo empresarial. La implementación de soluciones informáticas que realicen estas operaciones debido a diversos factores no comparte un formato o estándar, en pocas ocasiones mantienen relación, y sus desarrolladores implementan las mismas funcionalidades o los mismos procedimientos de manera distinta cada vez. Esta separación ha provocado no solo el incremento del número de soluciones sino además los gastos en tiempo y dinero para los entes interesados en obtener dichas herramientas.

La utilización de asistentes matemáticos para realizar algunas de estas operaciones es una práctica extendida. Dentro de los asistentes más utilizados se pueden mencionar Matlab, Octave, R, Mathematica, Derive, MAPLE, Excel entre otros. No todas estas herramientas son libres por lo que está limitado su uso o modificaciones bajo licencia. Además, condicionan su uso al conocimiento previo de los lenguajes particulares de cada uno de ellos. La utilización de una arquitectura de desarrollo única para las matemáticas crearía homogeneidad en las soluciones. La definición de un lenguaje de programación estandarizaría todos los códigos y permitiría la reutilización de los mismos. Es por ello que el departamento de Ciencias Básicas decidió establecer un marco de trabajo único para el conjunto de herramientas a desarrollar y que se describe a continuación.

Lenguaje de programación: Python 3.5

La decisión de utilizar Python como lenguaje de programación fue prácticamente unánime en la discusión con los miembros del departamento teniendo en cuenta las facilidades que brinda este lenguaje para el trabajo de las matemáticas. Python cuenta con bibliotecas especializadas como son:

- Pyomo: para el trabajo con optimización de funciones
- NumPy: para gestionar arreglos optimizados y el trabajo numérico
- SciPy: contiene herramientas y algoritmos matemáticos
- Matplotlib: permite la generación de gráficos a partir de datos contenidos en listas

Además, incluye por defecto la biblioteca *Math* que permite trabajar con funciones trigonométricas, creación de sucesiones aleatorias, redondeos, evaluación de funciones, entre otras opciones. A partir de la selección de este lenguaje, núcleo fundamental del marco de trabajo, el departamento de Ciencias Básicas de la Facultad 3 propone la siguiente línea tecnológica para el desarrollo de las herramientas de apoyo a la docencia:

Framework de desarrollo: Django 1.8.0

Django es un framework web de Python de alto nivel que fomenta el rápido desarrollo, diseño limpio y pragmático. Es construido por desarrolladores con experiencia, que se encarga de gran parte del desarrollo web para que los usuarios puedan centrarse en la escritura de su aplicación sin necesidad de reinventar la rueda. Es de código abierto y libre.

Esta línea tecnológica permite además incluir todas las facilidades de la web al desarrollo. Se escoge trabajar sobre la web con una tecnología que permita la ejecución de la herramienta en distintas estaciones de trabajo o en un servidor al cual tengan acceso los usuarios. El marco de trabajo permitirá una homogeneidad en todas las soluciones creadas por y para el departamento, apoyando el proceso de enseñanza aprendizaje y el trabajo metodológico en los colectivos de las asignaturas pertenecientes al departamento. La descripción de cómo son empleadas estas tecnologías en SIMUCI se detallan en el epígrafe 2.3.3 del capítulo 2.

Conocido el marco de trabajo que proponen los clientes de la herramienta a desarrollar resulta importante analizar y determinar una metodología de desarrollo de software que permita guiar la construcción de SIMUCI. El siguiente epígrafe se dedicará a la selección de dicha metodología a partir del estudio de los diferentes enfoques que existen y las necesidades de los clientes y la investigación.

1.5. Metodología para el desarrollo de software

Siendo conscientes de lo cambiante y amplio que es el mundo del software, una metodología debe ser lo suficientemente precisa como para que todas las personas involucradas en el

proceso la puedan seguir y sea de utilidad como pauta común. Pero también tiene que ser lo suficientemente adaptable, sencilla y completa como para aplicarse en distintos proyectos y que su utilización sea provechosa.

Hasta hace poco el proceso de desarrollo de software llevaba asociado un marcado énfasis en el control mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Si bien este esquema "tradicional" para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso, no resulta ser el más adecuado para la mayoría de los proyectos actuales, en que el entorno del sistema es muy cambiante, existan pocos roles y se necesite reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad. Ante las dificultades de utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir del “buen hacer” de la Ingeniería del Software, emergiendo como posible solución las metodologías ágiles (RODRÍGUEZ *et al.* 2012).

Las metodologías ágiles no tratan de predecir cómo será el proceso, sino adaptarlo a los cambios que puedan surgir en las etapas de desarrollo. Se centran en el factor humano y el producto software, dándole mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental con iteraciones muy cortas (DUARTE Y ROJAS 2008). Dentro del enfoque ágil existen varias metodologías que han demostrado su eficacia y éxito, dependiendo de las particularidades del proyecto. Conforme con (LETELIER 2006) se resumen a continuación sus principales características:

SCRUM: Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. La primera es que el desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

Crystal Methodologies: Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo, Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

Dynamic Systems Development Method (DSDM): Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.

Adaptive Software Development (ASD): Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda se desarrollan las características, finalmente en la tercera se revisa su calidad y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

Feature-Driven Development (FDD): Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.

Lean Development (LD): Definida por Bob Charette's a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la

productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios.

eXtreme Programming (XP): XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el 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. El ciclo de vida ideal de XP consta de cuatro fases: planeación, diseño, codificación y pruebas.

La Tabla 1, obtenida de (LETELIER 2006), compara las distintas aproximaciones ágiles en base a tres parámetros: vista del sistema como algo cambiante, colaboración entre los miembros del equipo y características más específicas de la propia metodología como son simplicidad, excelencia técnica, resultados, adaptabilidad y prácticas de colaboración. También incorpora como referencia no ágil el **Capability Madurity Model (CMM)**.

Tabla 1: Ranking de agilidad

	CMM	ASD	Crystal	DSDM	FDD	LD	Scrum	XP
Sistema como algo cambiante	1	5	4	3	3	4	5	5
Colaboración	2	5	5	4	4	4	5	5
Características Metodología (CM)								
-Resultados	2	5	5	4	4	4	5	5
-Simplicidad	1	4	4	3	5	3	5	5
-Adaptabilidad	2	5	5	3	3	4	4	3
-Excelencia técnica	4	3	3	4	4	4	3	4
-Prácticas de colaboración	2	5	5	4	3	3	4	5
Media CM	2.2	4.4	4.4	3.6	3.8	3.6	4.2	4.4
Media Total	1.7	4.8	4.5	3.6	3.6	3.9	4.7	4.8

Como se puede observar en la imagen anterior, existe una significativa diferencia del índice de agilidad entre todas las metodologías con respecto a CMM, destacándose entre ellas ASD, SCRUM y XP como las más ágiles. Cualquieras de estos tres enfoques comparten criterios en común que los hacen factibles para regir el presente trabajo, sin embargo, existen ciertos factores que potencian la elección de XP como metodología de desarrollo. Estos elementos son los siguientes:

- Equipo de desarrollo pequeño (3 personas) y con conocimientos previos de la metodología: XP está concebida para ser utilizada en proyectos que cuenten con un grupo muy reducido de desarrollo.
- El mayor peso de la investigación se centre en el código fuente como principal artefacto generado y no en la documentación: XP es descrita como una metodología de fácil prototipado y capaz generar pocos artefactos.
- Los requisitos de SIMUCI tienden a cambiar frecuentemente: XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existan un alto riesgo técnico.
- Frecuente integración del cliente con el desarrollador: XP potencia al máximo el trabajo en grupo. Tanto los clientes como los desarrolladores son parte del equipo y están involucrados en el desarrollo del software.
- Satisfacción del cliente: esta metodología trata de dar al cliente el software que necesita y cuando lo necesita, por lo que se debe responder rápido a sus necesidades, incluso cuando los cambios sean al final de ciclo de la programación.
- Dinamismo y flexibilidad para especificar requisitos: los requisitos del sistema son especificados a través de historias de usuario. El cliente describe brevemente las características que el sistema debe poseer en pequeños textos. La redacción de los mismos se realiza bajo la terminología del cliente, no del desarrollador, de forma que sea más clara, sencilla y sin profundizar en detalles.

1.6. Conclusiones del capítulo

El estudio realizado en el presente capítulo sobre el estado actual de los referentes teóricos del software educativo como medio de enseñanza, las características de la asignatura Simulación en la UCI, el marco de trabajo que propone el departamento de Ciencias Básicas de la Facultad 3 de la UCI, y la tendencia actual en el uso de metodologías ágiles en el desarrollo de software, permitió llegar a las siguientes conclusiones:

- El software educativo a desarrollar será concebido con un propósito específico: apoyar la labor del departamento de Ciencias Básicas de la Facultad 3 en el proceso de aprendizaje de los estudiantes.
- Se determinaron las funciones motivadora, instructiva e informativa como características del software educativo que deben incorporarse a la herramienta de apoyo a la asignatura Simulación resultante de la presente investigación, y que responden a la necesidad de elevar la motivación por el estudio de las ciencias básicas en los estudiantes de la UCI.
- La asignatura Simulación en la UCI, pese a sus características y contenidos, necesita del apoyo de una herramienta informática para disminuir el tiempo dedicado a cálculos auxiliares y permita visualizar, paso a paso, la ejecución de los métodos que integra.
- El marco de trabajo que propone el departamento de Ciencias Básicas de la Facultad 3 para desarrollar herramientas de apoyo a sus asignaturas, establece las principales tecnologías necesarias para implementar SIMUCI, propiciando homogeneidad en todas las soluciones que se rijan por este marco de trabajo.
- Con la metodología XP queda solventada la necesidad de conocer cómo guiar el proceso de desarrollo del software que propone la investigación y requieren los profesores del departamento de Ciencias Básicas.

CAPÍTULO II:

PROPUESTA DE SOLUCIÓN

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

2.1. Introducción

El actual capítulo persigue como objetivo describir los principales artefactos que son generados como consecuencia de la metodología de software empleada durante las fases de planeación y diseño, capaces de modelar el comportamiento de SIMUCI en cuanto a funciones, estructura, tecnología y buenas prácticas de desarrollo. Para ello, se procedió a la construcción de las historias de usuario descritas por los profesores del departamento de Ciencias Básicas de la Facultad 3 y a la definición de los requisitos funcionales y no funcionales de la aplicación. También se caracterizaron las herramientas y tecnologías que cumplimentan dichos requisitos, y se elaboraron en conjunto los planes de iteraciones, duración de iteraciones y de entregas. Además para describir la arquitectura de la aplicación se emplearon tarjetas CRC, el patrón arquitectónico Modelo – Plantilla – Vista y los patrones de diseño pertenecientes a la familia GRASP: controlador, bajo acoplamiento y alta cohesión. En un último momento se presentan las conclusiones del capítulo.

2.2. Descripción del negocio

Como se ha mencionado hasta el momento, la presente investigación pretende el diseño e implementación de un software educativo que apoye el PEA de la asignatura Simulación en la UCI. El desarrollo de la aplicación se encontrará regido por el marco de trabajo propuesto por el departamento de Ciencias Básicas de la Facultad 3 para el desarrollo de herramientas que apoyen sus asignaturas. SIMUCI deberá estar capacitada para la aplicación de los métodos que se imparten actualmente en la asignatura Simulación, potencializando el uso de ellos, enfocando el aprendizaje, a raíz de la rapidez de los cálculos relacionados a los métodos, a la interpretación de los resultados que se derivan de los mismos, así como a su comparación a partir de la ejecución de similares procedimientos. Además la herramienta, como software educativo, debe cumplir con las funciones motivadora, informativa e instructiva que se identificaron en el capítulo anterior.

SIMUCI, como muestra la **Figura 1**, debe brindar la posibilidad de ejecutar todos los métodos que propone paso a paso y de esta forma nutrir al usuario de los conocimientos básicos sobre el procedimiento estadístico, matemático y lógico aparejados a los propios métodos. La herramienta contendrá 4 módulos: números aleatorios, observaciones aleatorias, test de validación e intervalo de confianza, donde cada módulo contiene los métodos que se relacionan en el epígrafe 1.3. Es importante destacar que un usuario o actor del sistema será

considerado como cualquier persona interesada en utilizar la herramienta para la ejecución de alguno de los módulos que en ella se implementan.

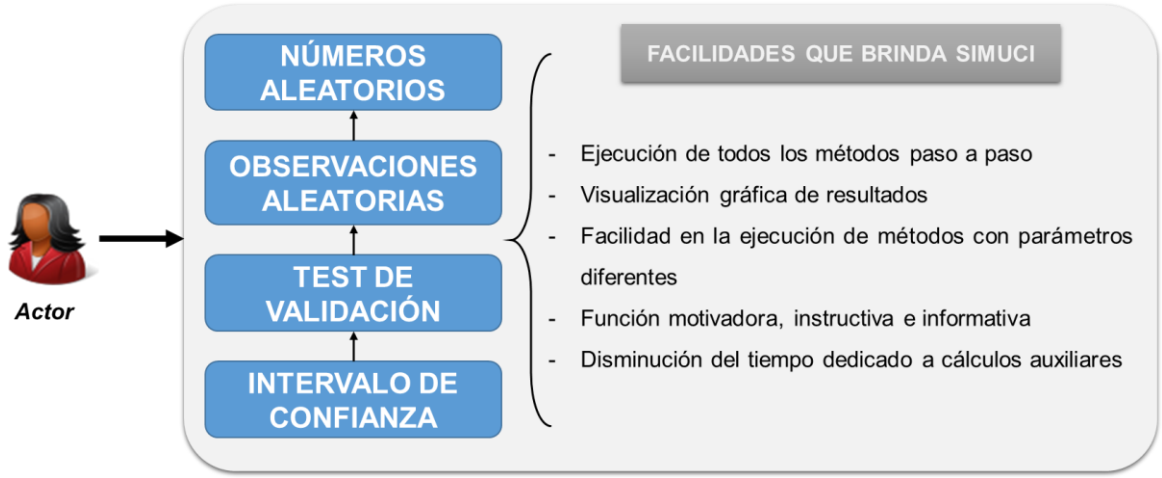


Figura 1: Esquema funcional de SIMUCI (elaboración propia)

Para lograr lo anterior los autores del presente trabajo dedicarán los siguientes epígrafes a la determinación de los aspectos necesarios para diseñar a SIMUCI y que la misma contemple calidad, robustez y eficiencia en las funcionalidades que debe brindar.

2.3. Planeación

La planeación de SIMUCI comenzó con la interacción entre los profesores del departamento de Ciencias Básicas de la Facultad 3 y el desarrollador, con el objetivo de descubrir los requisitos del sistema a través de las historias de usuario, las cuales sirvieron de guía para todo el proceso de desarrollo. Además, en dependencia de cómo fueron ordenadas y clasificadas según las necesidades de los clientes, se identificaron el número y tamaño de iteraciones a implementar, obteniéndose como resultado de esta fase, un plan de entregas donde se realizó una estimación de las versiones que tendrá el producto final.

2.3.1. Historias de usuario

Las historias de usuario (HU) constituyen el principal artefacto que se genera como parte de la puesta en práctica de esta fase de la metodología. Estas describen brevemente el comportamiento del sistema de forma dinámica y flexible, en cualquier momento pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU es considerada lo suficientemente comprensible y delimitada como

para que pueda ser implementada en pocas semanas (LETELIER 2006). La Tabla 2, obtenida de (BECK 2000), considerado como el padre de la filosofía XP, representa los campos principales que describen una HU.

Tabla 2: Descripción general de una HU

<table>
<tr><th colspan="3">HISTORIA DE USUARIO</th></tr>
<tr><td>No: Es un número único que se le asigna a cada HU con el fin de lograr una mejor organización de estas.</td><td>Iteración asignada: Define en que iteración del proceso de desarrollo de software será implementada la HU.</td><td>Nombre: Nombre de la HU. Debe ser descriptivo, en la medida de las posibilidades, de lo que se implementará y no muy extenso.</td></tr>
<tr><td colspan="2">Prioridad del negocio: Define la prioridad que presenta la HU para el negocio, puede ser: Alta, Media o Baja.</td><td>Nivel de complejidad: Define el nivel de complejidad que presenta la HU para su implementación, puede ser: Alta, Media o Baja.</td></tr>
<tr><td colspan="2">Punto de estimación: Estimación del esfuerzo asociado a la implementación de las historias. Un punto de estimación, equivale a una semana ideal de programación.</td><td>Riesgo en el desarrollo: Nivel de riesgo de la HU en el desarrollo de la aplicación. Puede ser Alto, Medio o Bajo,</td></tr>
<tr><td colspan="3">Descripción: Se describe el requerimiento que da origen a la HU. La descripción debe ser corta, precisa y dejar claro qué es lo que se desea hacer.</td></tr>
<tr><td colspan="3">Observaciones: Aspectos que se deben tener en cuenta para implementar la historia de usuario.</td></tr>
</table>

La Tabla 3 hace referencia a la HU correspondiente a la funcionalidad: generar números aleatorios utilizando el método congruencial mixto. El resto de las HU identificadas, pueden ser consultadas en los anexos del 1 al 15 del presente documento.

Tabla 3: Descripción de la HU1 (elaboración propia)

<table>
<tr><th colspan="3">HISTORIA DE USUARIO</th></tr>
<tr><td>No: 1</td><td>Iteración asignada: 1</td><td>Nombre: Generar números aleatorios utilizando el método congruencial mixto.</td></tr>
<tr><td colspan="2">Prioridad del negocio: Alta</td><td>Nivel de complejidad: Medio</td></tr>
<tr><td colspan="2">Punto de estimación: 1</td><td>Riesgo en el desarrollo: Medio</td></tr>
<tr><td colspan="3">Descripción: El sistema genera k números pseudoaleatorios a partir de los parámetros a, c, X₀, m y k.</td></tr>
<tr><td colspan="3">Observaciones:
Los campos correspondientes a los valores de los parámetros no pueden estar vacíos.
Los valores de los parámetros a, X₀ y c tienen que ser menor que el valor del parámetro m.
El valor del parámetro k tiene que ser menor o igual que el valor del parámetro m.</td></tr>
</table>

Los clientes de SIMUCI si bien no escribieron personalmente las HU, fueron quienes diseñaron su contenido y dirigieron su redacción. A pesar de lo anterior, el propósito que persiguen estas tarjetas no se vio afectado, conservando no solamente la terminología de los clientes, sino también su oficio como punto de partida en la planificación del proyecto. Desde el punto de vista del nivel de especificidad, se siguió la directiva de no profundizar ni en descripciones ni en procesos, siempre y cuando no fuera necesario, manteniéndolas de esta forma lo más breves y simples posibles. En cuanto a su codificación, se logró abstraer suficiente información de las mismas sin requerir en demasiadas aclaraciones por parte del departamento de Ciencias Básicas, siendo esto, un factor determinante para no ocasionar retrasos motivados por la falta de claridad en los requisitos.

2.3.2. Requisitos de software

Los requisitos del sistema constituyen un reflejo detallado de las necesidades del cliente, y la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto. Estos pueden ser agrupados en dos categorías: requisitos funcionales (RF) y requisitos no funcionales (RNF). Los RF describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa el ¿qué? y no el ¿cómo? se deben hacer esas transformaciones. Estos requisitos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema (CHAVES 2011). Por su parte los RNF tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento, interfaces de usuario, usabilidad, confiabilidad, mantenimiento, seguridad, portabilidad, estándares, etc. (CHAVES 2011).

A pesar de que los RF y RNF no forman parte de los artefactos que se generan como parte de la metodología XP, los autores de la actual investigación consideran que una descripción de estos podría facilitar al entendimiento del proceso de desarrollo. Para SIMUCI, fueron derivados los siguientes requisitos:

Requisitos funcionales

RF_1: Generar números aleatorios utilizando el método congruencial mixto.
RF_2: Generar números aleatorios utilizando el método congruencial aditivo.
RF_3: Generar números aleatorios utilizando el método congruencial multiplicativo.
RF_4: Generar números aleatorios utilizando generadores de Tausworthe.

RF_5: Generar observaciones de variables aleatorias utilizando el método de la transformada inversa para distribuciones conocidas.

RF_6: Generar observaciones de variables aleatorias utilizando el método de Box Müller y el teorema del límite central.

RF_7: Generar observaciones de variables aleatorias utilizando el método de convolución.

RF_8: Generar observaciones de variables aleatorias utilizando el método de composición.

RF_9: Validar la uniformidad de sucesiones de números aleatorios utilizando el test de Kolmogorov-Smirnov.

RF_{10}: Validar la uniformidad de sucesiones de números aleatorios utilizando el test de Chi Cuadrado.

RF_{11}: Validar la independencia en sucesiones de números aleatorios utilizando el test de las carreras por encima y por debajo de la media.

RF_{12}: Validar la independencia en sucesiones de números aleatorios utilizando el test de los huecos.

RF_{13}: Validar la independencia en sucesiones de números aleatorios utilizando el test de autocorrelación.

RF_{14}: Obtener el intervalo de confianza de una variable aleatoria a partir de sus observaciones.

RF_{15}: Guardar una lista de números aleatorios u observaciones aleatorias.

RF_{16}: Cargar una lista de números aleatorios u observaciones aleatorias.

Requisitos no funcionales

- **Apariencia o interfaz de usuario**
  - **RNF_1**: El sistema debe contar con un diseño sencillo, agradable, de fácil interacción e intuitivo, permitiendo que no sea necesario mucho entrenamiento o capacitación para su empleo.
  - **RNF_2**: Debe poseer un diseño adaptativo, es decir debe adaptarse a las dimensiones de la pantalla.
  - **RNF_3**: Los mensajes, títulos y demás textos que aparezcan en la interfaz del sistema deben aparecer en idioma español.
  - **RNF_4**: Las interfaces del sistema contendrán los datos de forma estructurada, permitiendo la correcta interpretación de la información.
  - **RNF_5**: La entrada incorrecta de datos será mostrada al usuario claramente, señalando los campos donde se encuentra el error.

- **Usabilidad**
  - **RNF$_6$**: El sistema estará desarrollado en una plataforma web, la cual permitirá al usuario tener acceso a la aplicación desde cualquier navegador.
  - **RNF$_7$**: El sistema debe cumplimentar con las funciones educativas: informativa, instructiva y motivadora.
- **Confiabilidad**
  - **RNF$_8$**: Los cálculos para la implementación de los modelos matemáticos deben estar en correspondencia con los estudiados, con el objetivo de garantizar que los resultados que se obtengan sean los esperados.
- **Hardware**
  - **RNF$_9$**: Para el cliente: como requisitos mínimos; un procesador Intel Pentium Dual Core a 1.6 GHz o AMD E-350 a 1.6GHz y 512MB de memoria RAM.
  - **RNF$_{10}$**: Para el servidor: como requisitos mínimos, un procesador Intel Pentium Dual Core a 1.6 GHz o AMD E-350 a 1.6GHz, capacidad para el disco duro de 5GB y 2GB de memoria RAM.
- **Software**
  - **RNF$_{11}$**: La aplicación deberá correr en los sistemas operativos Windows 7 o superior, o GNU Linux en cualquiera de sus distribuciones.
  - **RNF$_{12}$**: El navegador web que se utilice para interactuar con la aplicación deberá tener soporte para HTML5. Se recomienda utilizar Mozilla Firefox en su versión 30 o superior.
  - **RNF$_{13}$**: La computadora donde esté ejecutándose la aplicación, deberá tener instalado un intérprete de Python 3 y el módulo de Python para Django 1.8.0.
- **Disponibilidad**
  - **RNF$_{14}$**: El sistema deberá de encontrarse disponible en un servidor web para poder ser accedido por los estudiantes.

2.3.3. Línea base tecnológica

Luego de quedar reflejadas las distintas funcionalidades y características que de una forma u otra debe desempeñar la aplicación, resulta necesario la definición de una arquitectura tecnológica y aplicativa capaz de cumplimentar los RF y gestionar los RNF. Dicha línea, como consecuencia del marco de trabajo que propone el departamento de Ciencias Básicas de la Facultad 3, queda desglosada en las herramientas y tecnologías descritas a continuación.

2.3.3.1. Aplicación web

Una aplicación web es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (navegador, explorador o visualizador) como el servidor (servidor web) y el protocolo mediante el que se comunican (HTTP[1]) están estandarizados (MORA 2002). Al ser ejecutadas desde cualquier ordenador o dispositivo electrónico con una conexión estable a alguna red como Internet o Intranet, el usuario no necesita de grandes prestaciones para trabajar con ellas, consumiendo muy pocos recursos del servidor donde están instaladas. Las aplicaciones web son populares dentro del proceso de enseñanza-aprendizaje debido a lo práctico del navegador web como cliente ligero, a la independencia del sistema operativo, así como la facilidad para actualizar y mantener la aplicación sin distribuir e instalar software a miles de usuarios potenciales.

Es importante destacar que una página web puede contener elementos que permitan una comunicación activa entre el usuario y la información, evidenciado en la posibilitad de que el usuario acceda a los datos de modo interactivo. El desarrollo de SIMUCI como una aplicación web permitirá al estudiante un acceso rápido y centralizado a la gran mayoría de los métodos que se imparten actualmente en la asignatura Simulación, así como una elevada disponibilidad en cuanto a hora y lugar; podrá ser utilizada en cualquier momento o localidad, siempre que se establezca una conexión a la red y la aplicación se encuentre montada en un servidor.

2.3.3.2. Lenguaje de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana (MARK 2010). La elección de un lenguaje de programación está en correspondencia al tipo de aplicación que se pretenda desarrollar. La necesidad de concebir a SIMUCI como una aplicación web es solventada con la selección de Python en su versión 3.5 como lenguaje de programación del lado del servidor y de HTML5, en colaboración con CSS3 y JavaScript 5 como lenguajes del lado del cliente. Algunas características propias de estos lenguajes, así como útiles para el desarrollo de la aplicación en cuestión, quedan referidas a continuación.

[1] **HTTP:** Hypertext Transfer Protocol

Python es un lenguaje de programación interpretado, multiparadigma y multiplataforma, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible; usado ampliamente para el desarrollo de aplicaciones web (FERRER LLUCH 2016). Su crecimiento en el ámbito de la enseñanza es lento pero constante y viene confirmado por la progresiva aparición de proyectos científicos que lo utilizan como lenguaje de programación (NOGUERAS 2008). Las ventajas de utilizar Python para la codificación de estas herramientas que se proponen en el departamento de Ciencias Básicas de la Facultad 3, además de las planteadas en el epígrafe 1.4 del capítulo anterior, radican en la capacidad multiparadigma del lenguaje, es decir, al soportar tanto programación orientada a objetos, como funcional o imperativa, no obliga al programador a seguir un estilo de programación particular, sino que permite adaptarse a las características del sistema que será desarrollado, y en este caso además, a las asignaturas.

HTML5 es la nueva versión del lenguaje de marcado que se utiliza para estructurar páginas web. Surge como una evolución lógica de las especificaciones anteriores con los siguientes objetivos: separar totalmente la información y la forma de presentarla, resumir, simplificar y hacer más sencillo el código utilizado e incorporar nuevas etiquetas semánticas (DIEZ *et al.* 2012).

CSS es un lenguaje de estilos empleado para definir la presentación, el formato y la apariencia de un documento de marcaje que funcionan como espacios web. Las hojas de estilos nacen de la necesidad de diseñar la información de tal manera que se pueda separar el contenido de la presentación y, así, por una misma fuente de información, ofrecer diferentes presentaciones en función de dispositivos, servicios, contextos o aplicativos (PUIG 2013).

JavaScript, comúnmente abreviado JS, es un lenguaje de programación interpretado e implementado como parte de un navegador web (GAUCHAT 2012). Debido a que JS se ejecuta dentro del navegador de los clientes, puede ser utilizado para cambiar el aspecto de la pantalla en el dispositivo de los usuarios después que la página ha sido enviada por el servidor.

La integración de HTML5, CSS3 y JS en SIMUCI permitirá una mayor accesibilidad y limpieza en el código fuente, separar la estructura de las páginas web de su presentación, así como reducir la carga en el servidor e incrementar el rendimiento de la aplicación. También será capaz de brindarle al estudiante un entorno sencillo pero agradable, de fácil

interacción e intuitivo, donde la información esté contenida de forma estructurada, permitiendo la correcta interpretación de la misma.

2.3.3.3. Entorno de desarrollo integrado

Un entorno de desarrollo integrado, o más conocido por sus siglas en inglés IDE (Integrated Development Environment), es un entorno de programación que ha sido empaquetado como una aplicación y suele consistir en un editor de texto, un compilador, un depurador y una interfaz gráfica de usuario. Una de las ventajas de los IDE es que mientras el código es editado puede ser montado, y de esta forma el programa informa de posibles errores de sintaxis (RABADÁN UREÑA 2015).

En cuanto a distinción de IDEs, se pueden clasificar dependiendo del número de lenguajes a los que están dedicados. Existen muchos IDEs que soportan múltiples lenguajes, mientras otros están dedicados específicamente a un lenguaje de programación, permitiendo que las características sean lo más cercanas al paradigma de programación (RABADÁN UREÑA 2015). Tal es el caso de PyCharm, desarrollado por JetBrains. Constituye una herramienta profesional para el desarrollo de aplicaciones web basadas en Python, disponible para Windows, Mac y Linux. El proyecto de PyCharm está apoyado por una comunidad de desarrolladores dinámica y ofrece documentación y recursos de formación exhaustivos, así como una amplia selección de complementos de terceros (ROLDÁN ESTÉBANEZ 2015). Al ser Python precisamente el lenguaje de programación que se define como parte del marco de trabajo que propone el departamento de Ciencias Básicas de la Facultad 3 para el desarrollo de sus herramientas, y PyCharm, un IDE capaz de gestionar dicho lenguaje, además de la experiencia previa y conocimientos que tiene el desarrollador en el trabajo con esta herramienta, se propone su uso como entorno de programación para el desarrollo de la presente solución.

2.3.3.4. Framework de desarrollo web

El término framework es empleado en diferentes ámbitos del desarrollo de software, no solo en el de aplicaciones Web. Según (GUTIÉRREZ 2014) se puede considerar como una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo y reutilizar código ya existente. Un framework Web por tanto, según (GUTIÉRREZ 2014) también, puede ser definido como un conjunto de

componentes que conforman un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web. Supondría entonces una ventaja en cuanto a tiempo y buenas prácticas de desarrollo como son el uso de patrones arquitectónicos y de diseño, el empleo de un framework Web para la implementación de la solución de la presente investigación, solventando Django esta necesidad.

Django es un framework web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código. Se centra en automatizar todo lo posible y se adhiere al principio DRY[2] (ROLDÁN ESTÉBANEZ 2015). De acuerdo con (CONDORI AYALA 2012) impulsa el desarrollo de código limpio al utilizar una modificación del patrón arquitectónico Modelo - Vista - Controlador, llamado Modelo - Plantilla - Vista descrito con más especificidad en el epígrafe 2.4.2. Otro argumento válido para su descripción es la capacidad de manipular prácticas recomendables para el diseño de aplicaciones web, evidenciadas, en este caso, en el uso particular de patrones pertenecientes a la familia GRASP[3] como son alta cohesión, controlador y bajo acoplamiento. En el epígrafe 2.4.3 se realiza una explicación detallada de la presencia de dichos patrones en este framework seleccionado.

2.3.4. Estimación del esfuerzo por historia de usuario. Planes de iteraciones, duración de iteraciones y de entrega

En la metodología XP, el desarrollo del sistema es dividido en etapas para facilitar su realización. Por lo general, los proyectos de desarrollo constan de más de tres etapas, las cuales toman el nombre de iteraciones; de allí se obtiene el concepto de metodología iterativa, siendo la duración ideal de cada iteración entre una y tres semanas (ECHEVERRY TOBÓN Y DELGADO CARMONA 2007). Para cada iteración se define un módulo o conjuntos de historias que deben ser implementadas. Al final de cada iteración se obtiene como resultado la entrega del módulo correspondiente, el cual debe haber superado las pruebas de aceptación que establece el cliente para verificar el cumplimiento de los requisitos.

Aunque XP propone que el cliente es quien decida cuáles HU implementar y cuál es el grado de importancia de cada una en la correspondiente iteración, para SIMUCI, la tarea de escoger qué HU seleccionar fue realizada por el desarrollador, lo cual no generó problemas

[2] **DRY:** Don't Repeat Yourself
[3] **GRASP**: General Responsibility Assignment Software Patterns

en las entregas de los módulos funcionales. Para aproximar el tiempo que demoraría una HU se tomó como medida el punto. Un punto, equivale a una semana ideal de programación, lo que se traduce a cinco días de programación, dígase de lunes a viernes 8 horas diarias, para un total de 40 horas de programación.

En la Tabla 4 se pueden evidenciar las informaciones correspondientes a los planes de iteraciones, duración de iteraciones y de entregas, así como la estimación del esfuerzo obtenida por cada HU identificada e iteración implementada.

Tabla 4: Estimación del esfuerzo por HU. Planes de iteraciones, duración de iteraciones y de entrega (elaboración propia)

NO. DE ITERACIÓN	HISTORIA DE USUARIO	PRIORIDAD	PUNTOS DE ESTIMACIÓN		FIN DE ITERACIÓN
1	Generar números aleatorios utilizando el método congruencial mixto.	Alta	1	3	Cuarta Semana de Febrero
	Guardar lista de números aleatorios u observaciones aleatorias.	Alta			
	Generar números aleatorios utilizando el método congruencial aditivo.	Alta	1		
	Generar números aleatorios utilizando el método congruencial multiplicativo.	Alta			
	Generar números aleatorios utilizando generadores de Tausworthe.	Alta	1		
2	Cargar lista de números aleatorios u observaciones aleatorias.	Alta	2	3	Tercera Semana de Marzo
	Generar observaciones de variables aleatorias utilizando el método de la transformada inversa para distribuciones conocidas.	Alta			
	Generar observaciones de variables aleatorias utilizando el método de Box Müller y teorema del límite central.	Alta	1		
3	Generar observaciones de variables aleatorias utilizando el método de convolución.	Alta	2	3	

	Generar observaciones de variables aleatorias utilizando el método de composición.	Alta	1		Segunda Semana de Abril
4	Validar la uniformidad en sucesiones de números aleatorios utilizando el test de Kolmogorov-Smirnov.	Alta	1	3	Primera Semana de Mayo
	Validar la uniformidad en sucesiones de números utilizando el test de Chi Cuadrado.	Alta	1		
	Validar la independencia en sucesiones de números utilizando el test de las carreras por encima y por debajo de la media.	Alta	1		
5	Validar la independencia en sucesiones de números utilizando el test de los huecos.	Alta	1	3	Cuarta Semana de Mayo
	Validar la independencia en sucesiones de números utilizando el test de autocorrelación.	Alta	1		
	Obtener el intervalo de confianza de una variable aleatoria a partir de sus observaciones.	Alta	1		

2.4. Diseño

A diferencia de las metodologías pesadas, el diseño en XP se realiza durante todo el tiempo de vida del proyecto, siendo constantemente revisado y muy probablemente modificado debido a cambios presentados. La filosofía XP establece prácticas especializadas, que inciden directamente en la realización y elaboración del diseño de un software, no requiriendo que la representación del sistema sea mediante diagramas de clases basados en UML[4], sino que pueden emplearse indistintamente sencillos esquemas descritos en pizarras u otras técnicas como las tarjetas CRC[5] (ECHEVERRY TOBÓN Y DELGADO CARMONA 2007).

2.4.1. Tarjetas CRC

Las tarjetas CRC son el único artefacto de diseño que se genera como parte del proceso XP (GÓMEZ *et al.* 2014). Permiten visualizar el sistema en términos de objetos que componen

[4] **UML**: Unified Modeling Language
[5] **CRC:** Clase, Responsabilidad, Colaboración

la aplicación, identificando clases candidatas, responsabilidades y la forma que colaboran entre ellas (RIVERO *et al.* 1998).

El proceso de diseño de SIMUCI resultó iterativo, por lo cual, no se especificaron todas las tarjetas CRC durante la primera iteración. De forma general, al principio de cada iteración, fueron agregándose a las tarjetas existentes, responsabilidades y colaboraciones, o en caso de necesitarse, crear unas nuevas. El uso de estas tarjetas fue bastante útil, dando una idea clara de la arquitectura del sistema, distribución de clases y ubicación de las diferentes responsabilidades sobre la lógica del negocio. En total fueron especificadas 17 tarjetas CRC, que pueden ser consultadas en los anexos del 16 al 31. La siguiente tarjeta CRC (ver Tabla 5) corresponde a la clase controladora views.py, la cual implementa toda la lógica de negocio de la mayoría de los métodos relacionados con la asignatura Simulación.

Tabla 5: Descripción de la tarjeta CRC correspondiente a la clase views.py (elaboración propia)

TARJETA CRC	
Clase: views.py	
Responsabilidades:	**Colaboraciones**
Clase controladora que implementa los métodos que se imparten en la asignatura Simulación, así como sus validaciones. Encargada de renderizar los resultados obtenidos a las correspondientes plantillas.	Se relaciona con las clases: • urls.py • models.py • validators.py • gna_mixto.html • gna_multiplicativo.html • gna_aditivo.html • gna_tausworthe.html • tu_test_chi_cuadrado.html • tu_kolmogorov_smirnov.html • ti_carreras.html • ti_huecos.html • ti_autocorrelacion.html • goa_inversa.html • goa_normal.html • goa_uniforme.html • goa_composicion.html • goa_convolucion.html • intervalo_confianza.html

2.4.2. Patrón arquitectónico

(BUSCHMANN *et al.* 1996) expone los patrones arquitectónicos como una descripción de un problema particular y recurrente de diseño, que presentan un esquema genérico demostrado con éxito para su solución. El esquema de solución es especificado mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma en que colaboran entre sí. Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software y su uso debe ser estudiado con anticipación, con la intención de seleccionar el que mejor se adapte a los requisitos de calidad del sistema. La selección de un patrón arquitectónico es, por lo tanto, una decisión fundamental de diseño en el desarrollo de un sistema de software.

Modelo – Plantilla – Vista (MTV)

Django es basado en el conocido patrón arquitectónico Modelo – Vista – Controlador (MVC). En este patrón, el "*Modelo*" hace referencia al acceso a la capa de datos, la "*Vista*" describe la parte del sistema que selecciona qué mostrar y cómo mostrarlo, y el "*Controlador*" implica que componente decide qué vista usar, dependiendo de la entrada del usuario y accediendo al modelo si es necesario (CUMBA ARMIJOS Y BARRENO PILCO 2013). Someramente, la "*M*", "*V*" y "*C*" se separan en Django según (CONDORI AYALA 2012) de la siguiente manera:

- **M**, es la porción de acceso a los datos, manejada por la capa de datos de Django.
- **V**, es la porción que selecciona qué datos mostrar y cómo mostrarlos, manejada por la vista y las plantillas.
- **C**, es la porción que delega a la vista dependiendo de la entrada del usuario. Es manipulada por el propio framework siguiendo el URLConf y llamando a la función apropiada de Python para la URL[6] obtenida.

Debido que en Django la "C" es manejada por el propio framework y el comportamiento más importante se produce en los modelos, las plantillas y las vistas, es conocido como un framework MTV, donde:

- **M** significa *"Model"* (**Modelo**), la capa de acceso a los datos, contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el

[6] **URL**: Uniform Resource Locator

comportamiento que tienen, y las relaciones entre ellos. Esta capa permite gestionar las listas de números aleatorios que SIMUCI guardará o cargará en el sistema.

- **T** significa *"Template"* (**Plantilla**), la capa de presentación, contiene las decisiones relacionadas a como se presentan los datos obtenidos de la vista en el navegador web. Será utilizada para la presentación de los contenidos de la asignatura, los resultados de los métodos y para garantizar la comunicación entre los estudiantes y el ordenador.
- **V** significa *"View"* (**Vista**), la capa de la lógica de negocios, presenta en forma de funciones en Python. Su propósito es determinar qué datos serán visualizados. Puede pensarse como un puente entre el modelo y las plantillas. En SIMUCI será la capa encargada de implementar los métodos de la asignatura Simulación y ejecutarlos en dependencia de las necesidades del usuario.

Además, Django posee un mapeo de URLs que permite controlar el despliegue de las vistas. Esta configuración es conocida como URLConf. El trabajo del URLConf es leer la URL que el usuario solicitó, encontrar la vista apropiada para la solicitud y pasar cualquier variable que la vista necesite para completar su trabajo. El URLConf está construido con expresiones regulares en Python y sigue la filosofía: explícito es mejor que implícito. Este URLConf permite que las rutas que maneje Django sean agradables y entendibles para el usuario.

Una representación de este patrón definido en Django y haciendo uso del URLConf, en SIMUCI, lo constituye el siguiente ejemplo: un usuario desea validar que una sucesión de números aleatorios se encuentra distribuida uniformemente en el intervalo de [0;1) haciendo uso del test de Kolmogorov-Smirnov. La petición hecha a la aplicación por el usuario es representada en forma de URL “testu/kolmogorov_smirnov/”, interpretada por el URLConf y ubicada en la función correspondiente dentro de la clase controladora views.kolmogorov_smirnov, la cual a su vez hace uso del modelo load_numbers, que le permite cargar correctamente una sucesión de números aleatorios al sistema. El resultado obtenido de ejecutar la función views.kolmogorov_smirnov en la vista es renderizado[7] por la plantilla gna_tausworthe.html y presentado al usuario como respuesta a la petición realizada. La **Figura 2** ilustra este procedimiento.

[7] Renderizado: término usado en informática para referirse al proceso de generar una imagen desde un modelo.

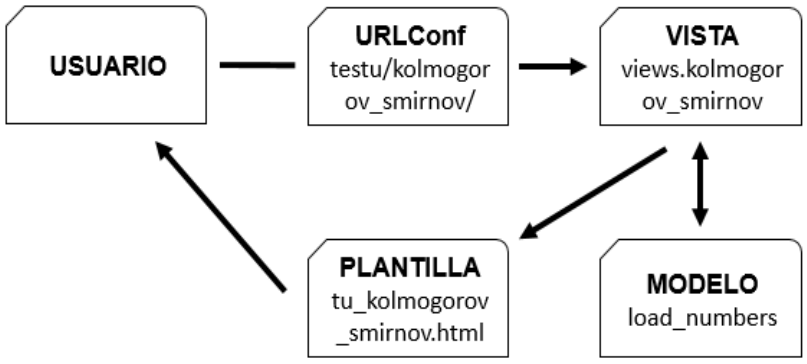


Figura 2: Ejemplo del patrón MPV en SIMUCI (elaboración propia)

2.4.3. Patrones de diseño

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular. Son menores en escala que los patrones arquitectónicos, y tienden a ser independientes de los lenguajes y paradigmas de programación. Su aplicación no tiene efectos en la estructura fundamental del sistema, pero sí sobre la de un subsistema, debido a que especifica a un mayor nivel de detalle, sin llegar a la implementación, el comportamiento de los componentes del subsistema (BUSCHMANN *et al.* 1996).

El empleo de patrones GRASP representa una ventaja para el desarrollo de software en cuanto a buenas prácticas de desarrollo. Patrones como controlador, alta cohesión y bajo acoplamiento son garantizados en SIMUCI con el empleo de Django como framework web. El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

En Django, el archivo views.py responde al patrón controlador, siendo el encargado de manejar qué datos serán presentados según las peticiones de los usuarios y no de qué forma. Sin embargo, es curioso destacar cómo con la existencia de una sola clase controladora, SIMUCI es capaz de mantener un bajo acoplamiento y aumentar la cohesión.

Principalmente se garantiza debido a que las peticiones de los usuarios son procesadas a través del archivo URLConf, donde cada URL definida se comporta como un mediador entre el usuario y la vista, y no directamente con la clase controladora, esta solo se limita a renderizar el resultado en la plantilla correspondiente como respuesta a la petición realizada, permitiendo así, que cada pieza de la aplicación que funciona sobre Django tenga un único propósito y puede ser modificada independientemente sin afectar las otras. Por ejemplo, un desarrollador puede cambiar la URL de cierta parte de la aplicación sin afectar la implementación subyacente, o un diseñador puede cambiar el HTML de una página sin tener que manipular el código Python que la renderiza. De igual forma, como cada pieza de SIMUCI tiene un único propósito, las responsabilidades de cada una van a estar estrechamente relacionadas.

2.4.4. Prototipos de interfaz de usuario

Un prototipo de interfaz de usuario constituye una representación previa de lo que será el software. En este se definen las vistas, los iconos, el estilo de las ventanas y menús, así como las acciones de la interfaz y los objetos. En el diseño de interfaces de usuario es muy importante realizar diferentes prototipos para comprobar de primera mano qué problemas habituales tiene el usuario cuando hace uso del sistema y poder así mejorar la interacción de éste con la aplicación. Para la construcción de los prototipos de SIMUCI se tuvieron en cuenta los requisitos no funcionales de apariencia, identificados previamente en el epígrafe 2.3.2. Como resultado de su seguimiento se obtienen prototipos de interfaces funcionales que cuentan con un diseño sencillo y agradable, de fácil interacción e intuitivos, donde los mensajes, títulos y demás textos que aparecen en el sistema se encuentran en español, y los datos están contenidos de forma estructurada garantizando la funcionalidad informativa que debía cumplir SIMUCI.

La **Figura 3** representa el prototipo de interfaz correspondiente a la página principal de SIMUCI, en la cual se puede evidenciar el menú izquierdo de navegación principal. Dicho menú será utilizado por el usuario para interactuar con los distintos módulos de la aplicación, que contienen la implementación de los métodos que se imparten en la asignatura Simulación. Además en la página principal, como en todas las demás secciones, el usuario será capaz de visualizar en la parte inferior, la información de los desarrolladores de la aplicación, una breve descripción del propósito que se persigue con la herramienta, y algunos enlaces a sitios que puedan ser considerados de interés en la universidad.

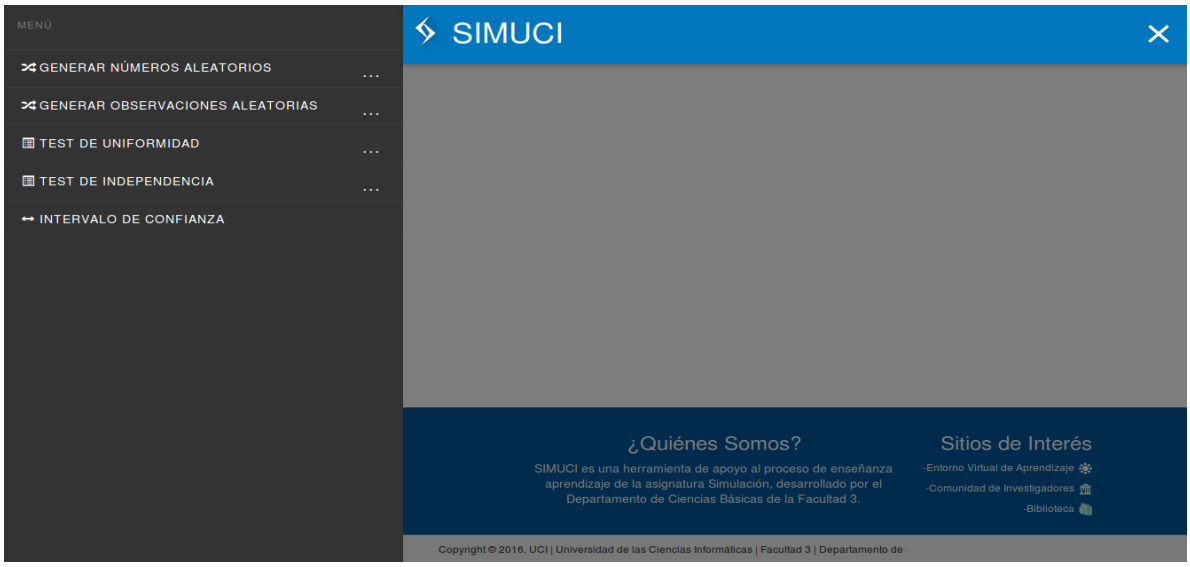


Figura 3: Prototipo de interfaz: pantalla principal con menú desplegado

Cuando se selecciona un módulo, como se observa en la **Figura 4**, se puede apreciar que la estructura de la página no sufre apenas modificaciones. En este contexto, se actualiza la etiqueta de la sección actual, la cual se encuentra ubicada a la derecha del nombre de la aplicación e indica la ruta en la que el usuario se encuentra. La información mostrada en cada apartado será la referente a la sección que se esté visualizando.

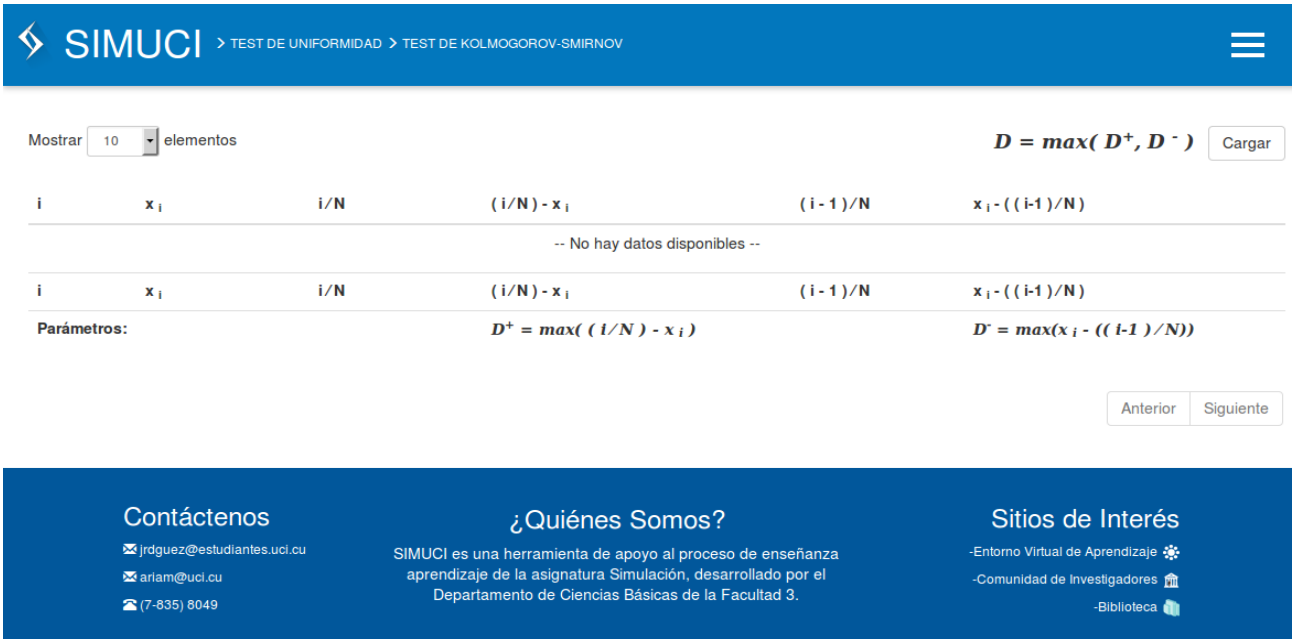


Figura 4: Prototipo de interfaz: Test de Kolmogorov-Smirnov

La construcción de prototipos en SIMUCI permitió reducir el riesgo de construir un producto software que no satisfaga las necesidades de los profesores del departamento de Ciencias Básicas de la Facultad 3. Además logró organizar inicialmente las ideas relacionadas con el diseño de la interfaz, adaptándose a los requisitos iniciales, para posteriormente desarrollar con mayor detalle la interfaz final.

2.5. Conclusiones del capítulo

El desarrollo de este capítulo dirigido a concretar una propuesta de solución con las características necesarias para cumplir con el objetivo de la presente investigación, permitió arrojar las siguientes conclusiones:

- La construcción de 16 HU y 19 tarjetas CRC constituyen artefactos necesarios propuestos por la metodología de software seleccionada que permitieron el entendimiento y la organización del proceso de desarrollo de SIMUCI.
- Derivadas de las HU, se realizó el levantamiento de 16 RF y 14 RNF, los cuales constituyeron un reflejo detallado de las necesidades del cliente.
- El lenguaje de programación Python, el framework de desarrollo Django junto a otras herramientas y tecnologías definidas como consecuencia del marco de trabajo del departamento de Ciencias Básicas de la Facultad 3 permitió implementar los RF y garantizar los RNF.
- Con el empleo de Django, como framework de desarrollo, se garantizan buenas prácticas para el desarrollo de SIMUCI, evidenciadas principalmente en el uso del patrón arquitectónico Modelo – Plantilla –Vista y en los patrones GRASP: controlador, bajo acoplamiento y alta cohesión.
- Las definiciones de los principales elementos en el capítulo, sentaron las bases para la construcción de SIMUCI y proceder a la siguiente fase de codificación.

CAPÍTULO III:

CODIFICACIÓN Y VALIDACIÓN DE LA SOLUCIÓN

CAPÍTULO III: CODIFICACIÓN Y VALIDACIÓN DE LA SOLUCIÓN

3.1. Introducción

El siguiente capítulo pretende describir las prácticas adoptadas por los autores del presente trabajo durante las dos últimas fases de desarrollo propuestas por la metodología de software seleccionada. En el mismo son descritos los principales estándares adoptados para la codificación del sistema, así como las pruebas de software realizadas a SIMUCI y los resultados obtenidos. Además, es empleado un cuestionario con el objetivo de determinar el grado de satisfacción de los usuarios para con la propuesta de solución siguiendo la técnica Iadov. En un último momento se presentan las conclusiones parciales del capítulo.

3.2. Codificación

La codificación es un proceso que se realiza de forma paralela al diseño y la cual está sujeta a varias observaciones por parte de XP. Entre las prácticas recomendadas durante esta fase según (ECHEVERRY TOBÓN Y DELGADO CARMONA 2007) se encuentran:

- El cliente debe estar siempre disponible, no solamente para solucionar las dudas del grupo de desarrollo, sino como parte de este.
- La comunicación entre el grupo de desarrollo es a través del código, con lo cual resulta indispensable seguir ciertos estándares de programación.
- Las Pruebas Unitarias son establecidas antes de escribir el código y ejecutadas constantemente ante cada modificación del sistema.

3.2.1. Estándares de codificación

XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que sigan ciertos estándares de codificación (ECHEVERRY TOBÓN Y DELGADO CARMONA 2007). Un estándar de codificación es un conjunto de reglas de notación y nomenclatura específicas para cada lenguaje de programación, que se usan y siguen durante la fase de implementación de un software. La aplicación de un estándar de programación al código fuente facilita el mantenimiento de un software, además de reducir el riesgo de que los desarrolladores introduzcan errores que no sean detectados por los compiladores, minimizando así el tiempo y coste de las actividades de depuración y pruebas necesarias para la detección y corrección de los mismos (CHARTE OJEDA 2002).

El programar siguiendo estándares es una práctica que no solo se recomienda en XP, sino que debe ser seguida en cualquier metodología de desarrollo. La codificación de SIMUCI fue realizada bajo las siguientes reglas (ver Tabla 6):

Tabla 6: Estándares de codificación utilizados en SIMUCI (elaboración propia)

ESTÁNDARES DE CODIFICACIÓN UTILIZADOS EN SIMUCI		
REGLA	**DESCRIPCIÓN**	**SIMUCI**
Máxima longitud de líneas	Las líneas de código son limitadas a un número máximo de caracteres.	Limitar todas las líneas de código (excepto los archivos **.js** y .**css**) a un máximo de 80 caracteres.
Líneas en blanco	Separar funcionalidades de alto nivel utilizando líneas en blanco.	Todas las funcionalidades de alto nivel serán separadas por dos líneas en blanco.
Espacios en blanco en expresiones y sentencias	Evitar el uso innecesario de espacios en blanco en determinadas situaciones.	Se evitará el uso de espacios en blanco en las siguientes situaciones: • Inmediatamente dentro de paréntesis, corchetes y llaves. • Inmediatamente antes de una coma, un punto y coma o dos puntos. • Inmediatamente antes del paréntesis que comienza la lista de argumentos de una función.
Comentarios en el código fuente	Durante el desarrollo de la aplicación resulta útil realizar comentarios explicativos que describan qué hace el código.	Los comentarios deben ser oraciones completas. Cada línea de un comentario comienza con el símbolo de numeral (#). Si un comentario es una frase u oración, su primera

		palabra debe comenzar con mayúscula. Si un comentario es corto puede omitirse el punto final.
Estilos de nombramiento	Se distingue los siguientes estilos de nombramiento: • minúscula (lowercase) • minúscula_con_guiones_bajos (lowercase_with_underscores) • MAYÚSCULA (UPPERCASE) • MAYÚSCULA_CON_GUIONES_BAJOS (UPPERCASE _WITH_UNDERSCORES) • PalabrasEnMayúscula (CamelCase) • minúsculaMayúscula (mixedCase)	Para el nombramiento de los métodos y declaración de variables se utilizó el estilo l**owercase**. En caso de ser la palabra compuesta es separada por guiones según el estilo **lowercase_with_underscores**. Además, se hizo empleo de un prefijo único delante de cada nombre capaz de identificar las funciones relacionadas a un mismo método.

3.2.2. Codificación de pruebas

En XP los programadores tienen el deber de programar las pruebas unitarias para cada módulo antes que el propio código de la aplicación. Esta práctica de acuerdo con (ECHEVERRY TOBÓN Y DELGADO CARMONA 2007) permite reducir considerablemente el tiempo dedicado a escribir determinado código si se identifican de antemano y de manera precisa cuáles son los casos especiales y rutas alternas que debe ser superar. XP recomienda para la concepción de pruebas unitarias, el uso de herramientas automatizadas en fases tempranas del desarrollo, de forma tal que puedan soportar una fase de pruebas continua y mantener organizados los casos de pruebas, permitiendo disminuir la ocurrencia de defectos y aprovechar las ventajas de la retroalimentación que se produce en el proceso.

La **Figura 5** representa el caso de prueba correspondiente al HU_9: validar uniformidad en sucesiones de números aleatorios utilizando el test de Kolmogorov-Smirnov. En dicha imagen se hace referencia a la clase kolmogorov_smirvov_test, la cual hereda de la clase TestCase, definida por Django para el desarrollo automatizado de pruebas unitarias. Los casos de pruebas son codificados en el archivo test.py dentro de la aplicación, haciendo uso del método setUp, encargado de crear los objetos capaces de probar los modelos definidos.

Una vez especificado el método setUp y los objetos a estudiar, se crean tantos métodos como se necesiten para evaluar los modelos. Un modelo incluye por lo general, la dirección URL que hace referencia a la función dentro de la vista (línea 112), un conjunto de parámetros de entradas definidos por el usuario (líneas 114 y 115) y las salidas que se esperan del sistema (línea 119).

```
106 class kolmogorov_smirnov_test(TestCase):
107     def setUp(self):
108         self.client = Client()
109 
110     def test_details(self):
111         response = self.client .post(
112             '/testu/kolmogorov_smirnov/',
113             {
114                 'cifras': '2',
115                 'numbers': '0.12,0.16,0.25,0.32,0.53,0.67',
116             }
117         )
118 
119         self.assertEqual(response.context['Dmas'], 0.35)
```

Figura 5: Caso de prueba correspondiente a la HU9 (elaboración propia)

Los casos de pruebas correspondientes a las restantes HU pueden ser consultados en los anexos del 32 al 47.

3.3. Pruebas de software

El desarrollo de pruebas representa una de las prácticas fundamentales en las cuales está basada la filosofía XP, siendo constantemente realizadas a largo del ciclo de vida del software (ECHEVERRY TOBÓN Y DELGADO CARMONA 2007). Esta fase en esencia describe un conjunto de actividades que tienen como premisa proporcionar información objetiva e independiente sobre la calidad del producto final, evaluando en el grado de lo posible, en qué medida el software cumple con las HU descritas por el cliente a través de un conjunto finito de casos de prueba debidamente seleccionados.

XP divide las pruebas del sistema en dos grandes grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores; y pruebas de aceptación, derivadas de las HU codificadas como parte de la liberación del software.

3.3.1. Pruebas unitarias

En los últimos años, los test unitarios han ido tomando cada vez más fuerza en el proceso de desarrollo de software, integrándose de una manera altamente productiva. Las pruebas unitarias representan una alternativa para encontrar y corregir la mayoría de los errores de implementación de un software (VIVAS *et al.* 2013). Conducen de manera rápida y sencilla a producir código de calidad mediante su codificación, como bien se describe en el epígrafe 3.1.2. El resultado que se obtiene luego de ejecutar estas pruebas es “OK” en caso de éxito y “FAIL” en caso contrario.

De manera general para SIMUCI fueron codificados 17 casos de prueba, de los cuales 13 resultaron satisfactorios, representando aproximadamente un 76% del total de casos, siendo el otro 24% inmediatamente tratado y solucionado luego de una revisión exhaustiva del código fuente del sistema. La **Figura 6** corresponde a la ejecución del caso de prueba correspondiente a la $HU_{9,}$ evidenciándose el pase satisfactorio de la misma.

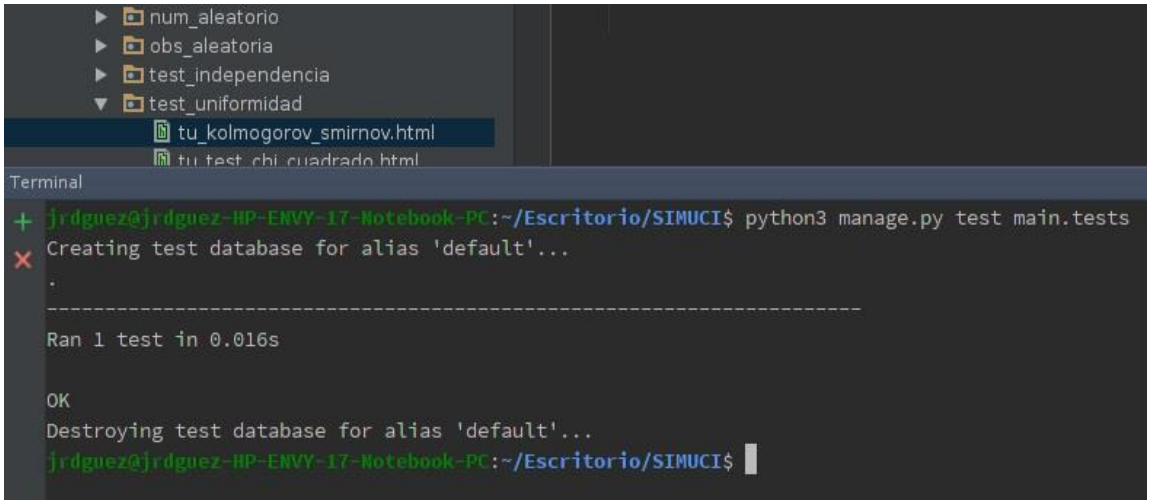


Figura 6: Pase satisfactorio del caso de prueba definido para la HU9 (elaboración propia)

3.3.2. Pruebas de aceptación

Las pruebas de aceptación son especificadas por el cliente y deben estar diseñadas con base en las HU. Se centran en las características y funcionalidades generales del sistema con el objetivo de corroborar si el comportamiento del software responde a las necesidades del cliente, representado con ello el fin de una iteración y el principio de otra (ECHEVERRY TOBÓN Y DELGADO CARMONA 2007).

Normalmente el cliente no es capaz de escribir las pruebas de aceptación por sí mismo, necesitando la ayuda de alguien más que pueda primero traducir sus juegos de datos a pruebas. La medida para la verificación de tales pruebas está basada en porcentajes. Con el tiempo se espera que el resultado de estas aumente en un valor próximo al 100%. Conforme se vaya cerrando una iteración, es necesario la clasificación de aquellas pruebas que fallan con el objetivo de elaborar un gráfico donde se muestre su progresión a lo largo del tiempo de vida del proyecto.

Para la realización de las pruebas se describieron casos de pruebas por cada HU y su correspondiente juego de datos a probar. Durante el desarrollo de este proceso se obtuvieron los siguientes porcentajes, que muestran el grado de satisfacción por parte del cliente en las diferentes iteraciones, evidenciados de manera general en la siguiente figura:

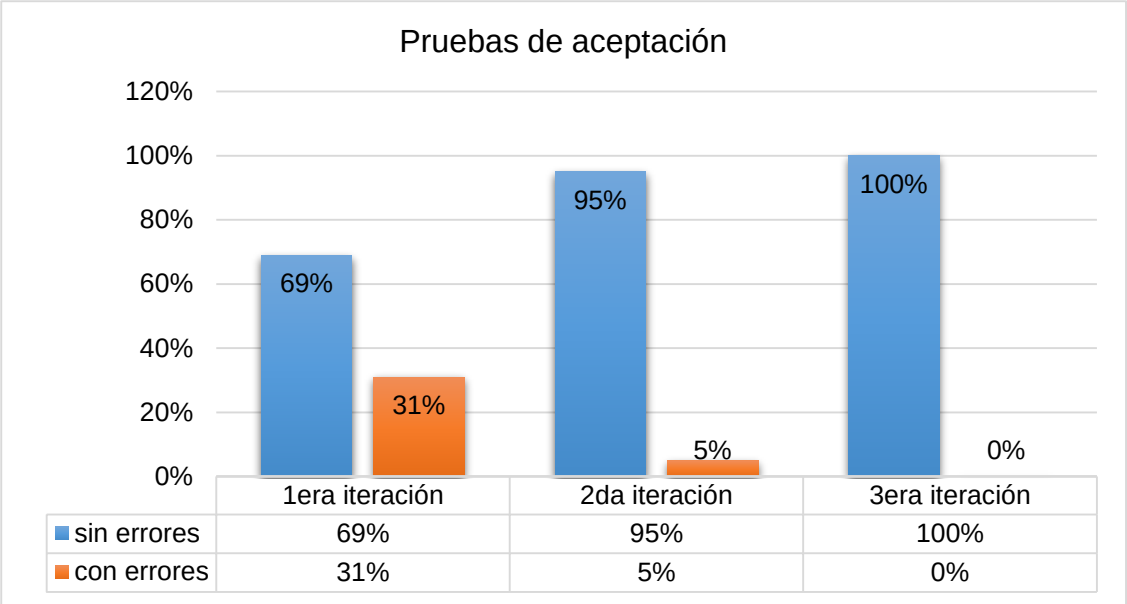


Figura 7: Resultados de las pruebas de aceptación (elaboración propia)

Como se puede observar en la primera iteración, de un total de 16 HU probadas, un 31% de ellas arrojaron no conformidades (NC). A raíz de este resultado se procedió a analizar exhaustivamente el código de la aplicación con el objetivo de intentar erradicar en su totalidad las deficiencias encontradas. Con el interés de comprobar si estas fueron suprimidas, se realizó una segunda iteración, donde se pudo evidenciar una disminución de hasta un 5% en los errores detectados. Finalmente, en una tercera iteración, luego de identificados y tratados las causas que arrojaban el porciento anterior, se obtuvo como resultado, la no existencia de nuevas conformidades, emitiéndose como constancia de ello,

una carta de aceptación por parte del departamento de Ciencias Básicas de la Facultad 3 en la que expresan su conformidad para con la herramienta implementada, la cual puede ser consultada en el Anexo 48. El tratamiento de las NC encontradas, en su mayoría validaciones de campos incorrectos, puede ser consultado en la **Figura 8**.

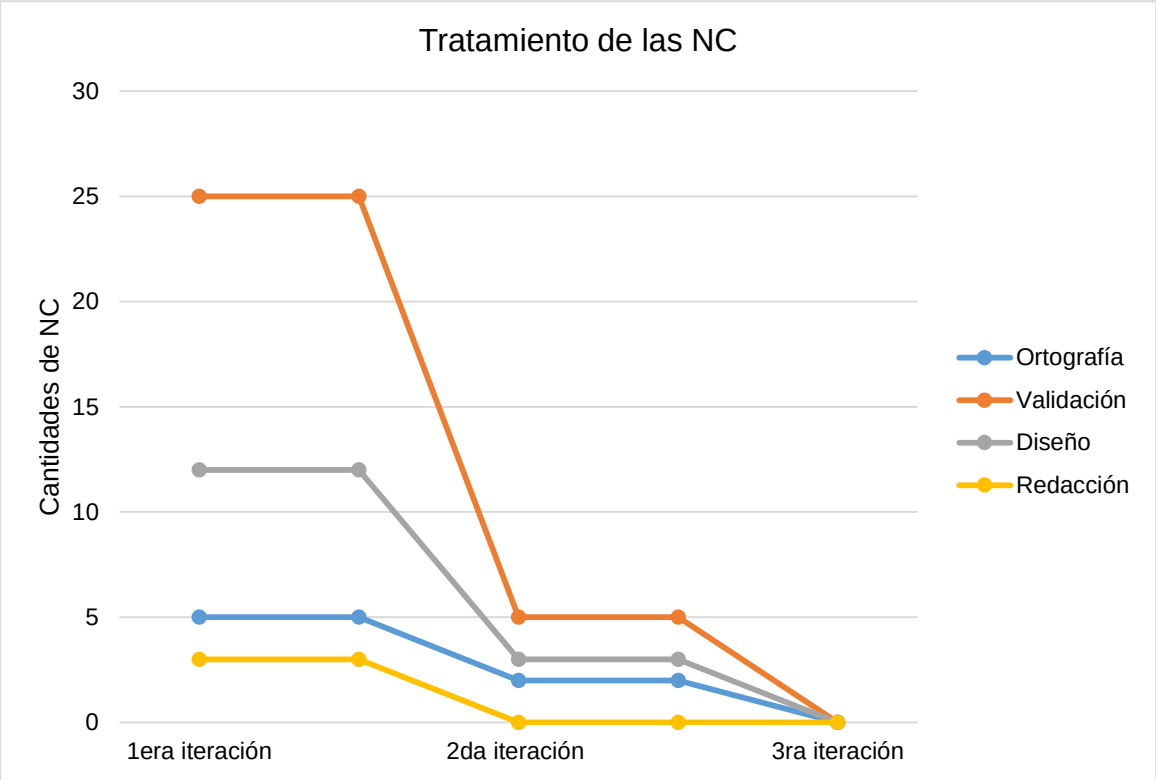


Figura 8: Tratamiento de las NC en las iteraciones (elaboración propia)

De forma similar, durante una revisión realizada a la herramienta en el grupo de calidad de CEGEL[8] de la Facultad 3, se identificaron un total de 9 NC. Como resultado de la no presentación de nuevas NC en una próxima iteración, fue firmada por del asesor de calidad de CEGEL: Ing. Yordanis García Leyva, una carta de liberación interna de productos software a favor de SIMUCI, demostrándose con ella la validez de la solución (Anexo 49).

Las Tabla 7 y 8 corresponden respectivamente al caso de prueba diseñado para la HU_9 y el juego de datos a probar. Los restantes casos de prueba pueden ser consultados en los anexos del 50 al 64.

[8] **CEGEL: Centro de Gobierno Electrónico**

Tabla 7: Diseño de caso de prueba de aceptación para la HU9 (elaboración propia)

CASO DE PRUEBA DE ACEPTACIÓN			
HISTORIA DE USUARIO		Validar la uniformidad de sucesiones de números aleatorios utilizando el test de Kolmogorov-Smirnov.	
DESCRIPCIÓN		Utilizar el test de Kolmogorov-Smirnov para validar que una sucesión de números aleatorios se encuentre distribuida uniformemente en el intervalo de 0 a 1.	
CONDICIONES DE EJECUCIÓN		Haber generado previamente un listado de números aleatorios.	
ESCENARIO DE PRUEBA		**FLUJO DEL ESCENARIO**	**RESULTADOS ESPERADOS**
EP1	El usuario introduce parámetros de entrada válidos para realizar los cálculos	1. Se presiona el botón cargar. 2. Se introducen los parámetros de entrada correctamente. 3. Se presiona el botón aceptar.	El sistema devuelve una tabla auxiliar donde se calcula el máximo de valor de D.
EP2	El usuario introduce parámetros de entrada inválidos para realizar los cálculos	1. Se presiona el botón cargar. 2. Se introducen los parámetros de entrada incorrectamente. 3. Se presiona el botón aceptar.	El sistema devuelve una excepción detallando el error cometido.

Tabla 8: Juego de datos para el caso de prueba de aceptación de la HU9 (elaboración propia)

JUEGOS DE DATOS DE PRUEBA			
ESCENARIO DE PRUEBA	**CANTIDAD DE CIFRAS SIGNIFICATIVAS**	**LISTADO DE NÚMEROS ALEATORIOS**	**RESPUESTA DEL SISTEMA**
EP1	2	0.12, 0.16, 0.25, 0.32, ,0.53, 0.67	D=0.35
EP2	X	0.12, 0.16, 0.25, 0.32, ,0.53, 0.67	ERROR: La cantidad de cifras significativas debe ser un valor numérico entero positivo.
EP2	0.2	0.12, 0.16, 0.25, 0.32, ,0.53, 0.67	ERROR: La cantidad de cifras significativas debe ser un valor numérico entero positivo.

EP2		0.12, 0.16, 0.25, 0.32, ,0.53, 0.67	ERROR: No pueden existir campos vacíos.
EP2	@	0.12, 0.16, 0.25, 0.32, ,0.53, 0.67	ERROR: La cantidad de cifras significativas debe ser un valor numérico entero positivo.
EP2	2	0.12, 0.16, 0.25, 0.32, ,0.53, 0.67, a, @	ERROR: El listado de números aleatorios debe ser valores numéricos comprendidos entre 0 y 1.

3.3.4. Validación de los resultados obtenidos

La técnica Iadov constituye una vía indirecta para el estudio de la satisfacción. Mediante esta se determina el nivel de satisfacción individual y grupal a partir de una encuesta elaborada y aplicada a una muestra seleccionada. Los criterios que se utilizan están fundamentados en tres preguntas cerradas que se intercalan dentro de un cuestionario y cuya relación el sujeto desconoce (LÓPEZ Y GONZÁLEZ 2002). Estas tres preguntas se encuentran relacionadas en la Tabla 9, a través de lo que se denomina el "**Cuadro Lógico de Iadov**, donde el número resultante de la interrelación indica la posición de cada sujeto en la escala de satisfacción.

Tabla 9: Cuadro Lógico de Iadov (elaboración propia)

P3: ¿La presentación de los resultados parciales de los métodos que ofrece SIMUCI ayuda a mejorar el entendimiento de los contenidos?	**P1:** ¿Disminuye el cúmulo de cálculos auxiliares la herramienta SIMUCI respecto a la realización manual en el aula?								
	Sí			No sé			No		
	P2: ¿Mejora la comprensión de los ejercicios, al disminuir los cálculos auxiliares?								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Clara satisfacción	**1**	**2**	**6**	**2**	**2**	**6**	**6**	**6**	**6**
Me satisfecho que insatisfecho	**2**	**2**	**3**	**2**	**3**	**3**	**6**	**3**	**6**
No definida	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
Me insatisfecho que satisfecho	**6**	**3**	**6**	**3**	**4**	**4**	**3**	**4**	**4**
Clara insatisfacción	**6**	**6**	**6**	**6**	**4**	**4**	**6**	**4**	**5**
No puedo decir	**2**	**3**	**6**	**3**	**3**	**3**	**6**	**3**	**4**

Para conocer el índice de satisfacción de los usuarios para con la herramienta desarrollada, se procede a diseñar un caso de estudio que pretende analizar el grado de satisfacción que experimentan los estudiantes del 4to año de la carrera de la Facultad 3 luego de la puesta en práctica de un examen de Simulación, visto desde dos perspectivas: primero de forma manual y después con el empleo de la herramienta informatizada. En total fueron encuestados 46 estudiantes, los cuales representan una muestra de aproximadamente un 82% de la población, tomándose como población los 56 estudiantes pertenecientes al cuarto año de la Facultad 3. Los resultados individuales de satisfacción se resumen en la **Figura 9**.

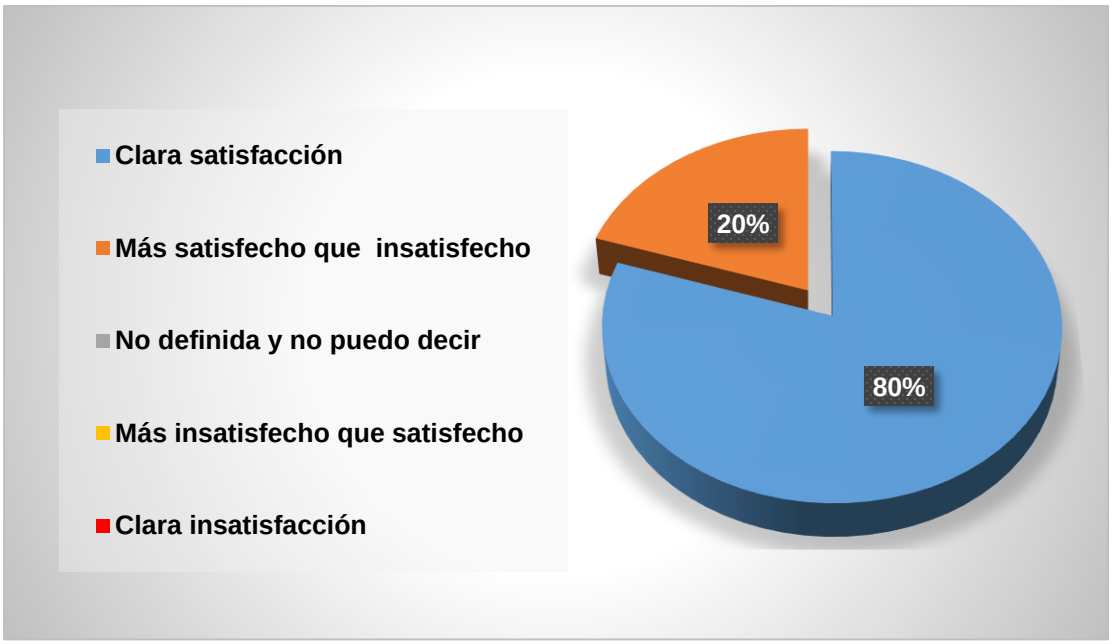


Figura 9: Resumen de los resultados individuales obtenidos (elaboración propia)

Para obtener el índice de satisfacción grupal (ISG) se parte de asociar los diferentes niveles de satisfacción individual de los encuestados en una escala numérica que oscila entre +1 y -1, donde los valores comprendidos entre -1 y -0,5 indican insatisfacción; entre -0,49 y +0,49 evidencian contradicción y entre 0,5 y 1 demuestran satisfacción. La tabla que se presenta a continuación (ver Tabla 10) establece la relación existente entre los resultados obtenidos en la satisfacción individual y su correspondiente valor en la escala de satisfacción.

Tabla 10: Relación entre la satisfacción individual y la escala de satisfacción (elaboración propia)

Escala	Significado	Satisfacción individual	%
+1	Clara satisfacción	37	80
+0.5	Más satisfecho que insatisfecho	9	20
0	No definida	0	0
-0.5	Más insatisfecho que satisfecho	0	0
-1	Clara insatisfacción	0	0

La fórmula utilizada para obtener el índice de satisfacción grupal fue:

ISG= a (+ 1) + b (0.5) + c (0) + d (- 0.5) + e (-1) / N

Donde a, b, c, d, e son las cantidades de alumnos clasificados en cada una de las escalas de satisfacción respectivamente y N es la cantidad de estudiantes tomados como muestra; siendo en este caso ISG = 0,90 aproximadamente. Como se puede apreciar el valor del índice es alto, lo que refleja satisfacción por parte de los estudiantes, aceptación de la propuesta y reconocimiento de su utilidad.

La técnica de Iadov contempla además dos preguntas complementarias de carácter abierto que permitieron profundizar en la naturaleza de las causas que originaron los diferentes niveles de satisfacción. Resultó significativo en el análisis de estas opiniones, la preponderancia de aspectos positivos que plantearon los estudiantes con respecto al uso de SIMUCI, lo cual sirve como fundamento del alto valor obtenido en el ISG.

3.4. Conclusiones del capítulo

Con la culminación de este capítulo dedicado a la codificación y validación de la propuesta de solución se obtuvieron los siguientes resultados:

- El uso de estándares de programación para gestionar aspectos como la máxima longitud de líneas, líneas en blanco, espacios en blanco en expresiones y sentencias, comentarios en el código fuente y estilos de nombramiento, permitieron mejorar la calidad y legibilidad del código de la aplicación, así como favorecer a un futuro mantenimiento.
- La codificación temprana de pruebas permitieron agilizar el proceso de codificación de SIMUCI a partir de la identificación de los casos especiales y rutas alternas que los métodos debían superar.

- Las pruebas de unidad, enfocadas en verificar el correcto funcionamiento de cada método arrojaron resultados satisfactorios y esperados. Por su parte las pruebas de aceptación validaron que el sistema desarrollado responde y satisface en un 100% de conformidad las necesidades de los profesores del departamento de Ciencias Básicas de la Facultad 3.
- La encuesta realizada a una muestra de 46 estudiantes del cuarto año de la Facultad 3 arrojo resultados positivos, reflejando satisfacción para con la herramienta implementada con un índice de 0,90.
- Todo lo anterior permitió validar la solución propuesta dando lugar al cumplimiento del objetivo general de la investigación.

CONCLUSIONES FINALES

Sobre la base del análisis, interpretación y sistematización de las investigaciones teóricas y empíricas, a continuación se presentan las conclusiones de la investigación:

1. El establecimiento del marco teórico de la investigación permitió demostrar la necesidad de desarrollar un software educativo que contribuya a la visualización numérica de los métodos que se imparten en la asignatura Simulación en la UCI e incorpore funciones motivadoras, informativas e innovadoras.

2. La metodología de software seleccionada, logró guiar satisfactoriamente el proceso de desarrollo de SIMUCI a través de sus fases de planeación, diseño, codificación y pruebas.

3. La realización de pruebas de software correspondientes a la metodología de desarrollo seleccionada, posibilitó validar las funcionalidades de SIMUCI. La aplicación de un cuestionario utilizando la técnica Iadov permitió medir la satisfacción de los usuarios para con la herramienta implementada. Ambos elementos ayudaron a catalogar como satisfactoria la propuesta de solución, dando lugar a que se cumpla en su totalidad con el objetivo general de la investigación.

RECOMENDACIONES

A pesar de que la herramienta desarrollada a partir de la siguiente investigación cumple con el objetivo general de la misma, en función de continuar su perfeccionamiento se recomienda:

- Incorporar a SIMUCI un módulo capaz de gestionar la tabla de Simulación para procesos determinados, permitiendo a los estudiantes ejecutar numerosas corridas de un mismo sistema y obtener resultados más precisos y confiables en las simulaciones que realicen.

- Implementar en SIMUCI un reconocedor de funciones para gestionar las variables aleatorias que siguen distribuciones no conocidas y elevar el alcance en el trabajo con la herramienta.

GLOSARIO DE TÉRMINOS

HTTP: Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto)

DRY: Don't Repeat Yourself (Una y solo una vez)

GRASP: General Responsibility Assignment Software Patterns (Patrones de software para asignar responsabilidades)

UML: Unified Modeling Language (Lenguaje Unificado de Modelado)

CRC: Clase, Responsabilidad, Colaboración

URL: Uniform Resource Locator (Localizador de Recursos Uniforme)

CEGEL: Centro de Gobierno Electrónico

Renderizado: término usado en informática para referirse al proceso de generar una imagen desde un modelo.

REFERENCIAS BIBLIOGRÁFICAS

1. BECK, K. Extreme programming explained: embrace change. addison-wesley professional, 2000. p. 0201616416

2. BUSCHMANN, F.; R. MEUNIER, et al. Pattern-oriented software architecture: a system of patterns, John Wiley & Sons Inc., New York, NY, 1996.

3. COLLAGUAZO, Q. and W. ALEJANDRO Software educativo en el proceso de enseñanza aprendizaje de los instrumentos de medida de precisión en los estudiantes del tercer año de bachillerato técnico de la Unidad Educativa Mushuc Pacari de la ciudad de Quito en el período 2014-2015, 2016.

4. CONDORI AYALA, J. L. Phython-DjangoFramework de desarrollo web para perfeccionistasBasado en el Modelo MTV Revista de Información, Tecnología y Sociedad, 2012: 36.

5. CUMBA ARMIJOS, P. D. and B. A. BARRENO PILCO Análisis de PYTHON con Django frente a Ruby on Rails para desarrollo ágil de aplicaciones web. Caso práctico: DECH, 2013.

6. CHARTE OJEDA, F. Visual C# .NET Guía práctica para usuarios, Anaya Multimedia, 2002.

7. CHAVES, M. A. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software InterSedes, 2011, 6(10).

8. DIEZ, T.; M. DOMINGUEZ, et al. Creación de páginas web accesibles con HTML5 Consultado el, 2012, 26.

9. DUARTE, A. O. and M. ROJAS Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo Avances en Sistemas e Informática, 2008, 5(2).

10. ECHEVERRY TOBÓN, L. M. and L. E. DELGADO CARMONA Caso práctico de la metodología ágil XP al desarrollo de software, 2007.

11. FERRER LLUCH, Z. Implementación del juego de las damas en un robot NAO, 2016. p.

12. GARCÍA, D. F. V. Software Educativo para el aprendizaje creativo del curso "Embriología Comparada" Revista Electrónica Educare, 2011, 15(2): 141-161.

13. GAUCHAT, J. D. El gran libro de HTML5, CSS3 y Javascript. Marcombo, 2012. p. 8426717829

14. GÓMEZ, A. R.; A. Q. DUARTE, et al. Desarrollo ágil de software aplicando programación extrema Revista Ingenio UFPSO, 2014, 5(1): 24-29.

15. GUTIÉRREZ, J. J. ¿ Qué es un framework Web? Available in: http://www. lsi. us. es/~ javierj/investigacion_ficheros/Framework. pdf Accessed May, 2014, 12.

16. HERNÁNDEZ SAMPIERI, R.; C. FERNÁNDEZ COLLADO, et al. Metodología de la investigación México: Editorial Mc Graw Hill, 2010.

17. LABAÑINO RIZO, C. Colección El Navegante La Habana: Dirección Nacional de Computación, 2005.

18. LABRADA, S. M. El Software Educativo un medio de enseñanza eficiente Cuadernos de Educación y Desarrollo, 2011, (29).

19. LAVIÑA ORUETA, J. and L. MENGUAL PAVÓN. Libro blanco de la Universidad digital. España, 2010.

20. LETELIER, P. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP), 2006.

21. LÓPEZ, A. and V. GONZÁLEZ La técnica de Iadov. Una aplicación para el estudio de la satisfacción de los alumnos por las clases de Educación Física Lecturas Educación Física y Deportes, Revista Digital, 2002, 47.

22. MARK. Learning Python, Fourth Edition, 2010.

23. MORA, S. L. Programación de aplicaciones web: historia, principios básicos y clientes web. Editorial Club Universitario, 2002. p. 8484542068

24. NOGUERAS, G. B. Python como entorno de desarrollo científico, Julio, 2008.

25. PUIG, J. C. CSS3 y Javascript avanzado, Barcelona: UOC, 2013.

26. RABADÁN UREÑA, A. Análisis de imágenes del rostro humano, 2015.

27. RAMOS, J. L. B. Los medios de enseñanza: clasificación, selección y aplicación Pixel-Bit: Revista de medios y educación, 2004, (24): 113-124.

28. RIVERO, L. C.; J. H. DOORN, et al. Una Estrategia de Análisis Orientada a Objetos basada en Escenarios: Aplicación en un Caso Real. WER, 1998. 79-88 p.

29. RODRÍGUEZ, G.; A. SORIA, et al. Supporting virtual meetings in distributed scrum teams Latin America Transactions, IEEE (Revista IEEE America Latina), 2012, 10(6): 2316-2323.

30. RODRÍGUEZ LAMAS, R.; M. GARCÍA VEGA, et al. Introducción a la informática educativa Ciudad de la Habana: ISPJAE, 2000.

31. ROLDÁN ESTÉBANEZ, D. Sistema multiplataforma de gestión integral de huertos urbanos, 2015.

32. SÁNCHEZ, J.; P. IRIARTE, et al. Construyendo y aprendiendo con el computador. Integración de medios interactivos para la capacitación de profesores en informática educativa. VIII Congreso Nacional de Informática Educativa. Universidad del Bio, 1999. 25-36 p.

33. SÁNCHEZ, T. and E. MARCELO Diseño y aplicación de un software educativo para desarrollar destrezas con criterio de desempeño del área de Matemática en los estudiantes de quinto año de Educación General Básica de la Unidad Educativa Cristiana Emanuel, de la ciudad de Macas, durante el año lectivo 2012-2013, 2013.

34. SARIOL FERNÁNDEZ, M. MATHNUM, Herramienta de apoyo a la asignatura Matemática IV. La Habana, 2015. 79.

35. UCI. Universidad de las Ciencias Informáticas: Historia., 2012.

36. VAN DER LINDE, G. ¿ Por qué es importante la interdisciplinariedad en la educación superior? Cuaderno de Pedagogía Universitaria, 2014, 4(8): 11-12.

37. VIVAS, L.; M. CAMBARIERI, et al. Un marco de trabajo para la Integración de Arquitecturas de Software con Metodologías Ágiles de Desarrollo. XVIII Congreso Argentino de Ciencias de la Computación, 2013. p.

ANEXOS

Anexo 1: Descripción de la HU "Generar números aleatorios utilizando el método congruencial aditivo"

HISTORIA DE USUARIO		
No: 2	**Iteración asignada:** 1	**Nombre:** Generar números aleatorios utilizando el método congruencial aditivo.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema genera ***k*** números pseudoaleatorios a partir de una lista de números aleatorios enteros y a partir de los parámetros de ***i, j, m*** y ***k.***		
Observaciones: • Los campos correspondientes a los valores de los parámetros no pueden estar vacíos. • El valor ***j*** tiene que ser mayor que el valor del parámetro ***i.*** • El valor del parámetro ***k*** tiene que ser menor o igual que el valor del parámetro ***m.***		

Anexo 2: Descripción de la HU "Generar números aleatorios utilizando el método congruencial multiplicativo"

HISTORIA DE USUARIO		
No: 3	**Iteración asignada:** 1	**Nombre:** Generar números aleatorios utilizando el método congruencial multiplicativo.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema genera ***k*** números pseudoaleatorios a partir de los parámetros ***a***, ***c***, X_0, ***m*** y ***k***.		
Observaciones: • Los campos correspondientes a los valores de los parámetros no pueden estar vacíos. • Los valores de los parámetros ***a*** y X_0 tienen que ser menor que el valor del parámetro ***m***. • El valor del parámetro ***k*** tiene que ser menor o igual que el valor del parámetro ***m.*** • El valor del parámetro ***c*** tiene que ser igual a cero.		

Anexo 3: Descripción de la HU "Generar números aleatorios utilizando generadores de Tausworthe"

HISTORIA DE USUARIO		
No: 4	**Iteración asignada:** 1	**Nombre:** Generar números aleatorios utilizando generadores de Tausworthe.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema genera ***k*** números pseudoaleatorios a partir de una secuencia de bits distinta de cero y los parámetros ***r***, ***q***.		
Observaciones: • Los campos correspondientes a los valores de los parámetros no pueden estar vacíos. • El valor ***q*** tiene que ser mayor que el valor del parámetro ***r.***		

Anexo 4: Descripción de la HU "Generar observaciones de variables aleatorias utilizando el método de la transformada inversa para distribuciones conocidas"

HISTORIA DE USUARIO		
No: 5	**Iteración asignada:** 2	**Nombre:** Generar observaciones de variables aleatorias utilizando el método de la transformada inversa para distribuciones conocidas.
Prioridad del negocio: Alta		**Nivel de complejidad:** Alta
Punto de estimación: 2		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema genera ***k*** observaciones aleatorias en dependencia de la distribución conocida seleccionada.		
Observaciones: El sistema debe cargar previamente una lista de números aleatorios.		

Anexo 5: Descripción de la HU "Generar observaciones de variables aleatorias utilizando el método de Box Müller y teorema del límite central"

HISTORIA DE USUARIO		
No: 6	**Iteración asignada:** 2	**Nombre:** Generar observaciones de variables aleatorias utilizando el método de Box Müller y teorema del límite central.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema genera ***k*** observaciones aleatorias en dependencia del método seleccionado.		
Observaciones: El sistema debe cargar previamente una lista de números aleatorios.		

Anexo 6: Descripción de la HU “Generar observaciones de variables aleatorias utilizando el método de convolución”

HISTORIA DE USUARIO		
No: 7	**Iteración asignada:** 3	**Nombre:** Generar observaciones de variables aleatorias utilizando el método de convolución.
Prioridad del negocio: Alta		**Nivel de complejidad:** Alto
Punto de estimación: 2		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema genera ***k*** observaciones aleatorias como la suma del resultado de varias distribuciones conocidas.		
Observaciones: El sistema debe cargar previamente una lista de números aleatorios.		

Anexo 7: Descripción de la HU “Generar observaciones de variables aleatorias utilizando el método de composición”

HISTORIA DE USUARIO		
No: 8	**Iteración asignada:** 3	**Nombre:** Generar observaciones de variables aleatorias utilizando el método de composición.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema genera ***k*** observaciones aleatorias a partir de la selección de varias distribuciones conocidas.		
Observaciones: El sistema debe cargar previamente una lista de números aleatorios.		

Anexo 8: Descripción de la HU “Validar la uniformidad en sucesiones de números aleatorios utilizando el test de Kolmogorov-Smirnov”

HISTORIA DE USUARIO		
No: 9	**Iteración asignada:** 4	**Nombre:** Validar la uniformidad en sucesiones de números aleatorios utilizando el test de Kolmogorov-Smirnov.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema devuelve como resultado el valor del estadígrafo D.		
Observaciones: El sistema debe cargar previamente una lista de números aleatorios.		

Anexo 9: Descripción de la HU "Validar la uniformidad en sucesiones de números utilizando el test de Chi Cuadrado"

HISTORIA DE USUARIO		
No: 10	**Iteración asignada:** 4	**Nombre:** Validar la uniformidad en sucesiones de números utilizando el test de Chi Cuadrado.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema devuelve como resultado el valor del estadígrafo X^2.		
Observaciones: El sistema debe cargar previamente una lista de números aleatorios.		

Anexo 10: Descripción de la HU "Validar la independencia en sucesiones de números utilizando el test de las carreras por encima y por debajo de la media"

HISTORIA DE USUARIO		
No: 11	**Iteración asignada:** 4	**Nombre**: Validar la independencia en sucesiones de números utilizando el test de las carreras por encima y por debajo de la media.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema devuelve como resultado el valor del estadígrafo Z.		
Observaciones: El sistema debe cargar previamente una lista de números aleatorios.		

Anexo 11: Descripción de la HU "Validar la independencia en sucesiones de números utilizando el test de los huecos"

HISTORIA DE USUARIO		
No: 12	**Iteración asignada:** 5	**Nombre:** Validar la independencia en sucesiones de números utilizando el test de los huecos.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema devuelve como resultado el valor del estadígrafo D.		
Observaciones: El sistema debe cargar previamente una lista de números aleatorios.		

Anexo 12: Descripción de la HU "Validar la independencia en sucesiones de números utilizando el test de autocorrelación"

HISTORIA DE USUARIO		
No: 13	**Iteración asignada:** 5	**Nombre:** Validar la independencia en sucesiones de números utilizando el test de autocorrelación.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema devuelve como resultado el valor del estadígrafo Z a partir de un paso ***m*** y una posición ***i***.		
Observaciones: • Los campos correspondientes a los valores de los parámetros no pueden estar vacíos. • El valor de ***m*** tiene que ser mayor que cero.		

Anexo 13: Descripción de la HU "Obtener el intervalo de confianza de una variable aleatoria a partir de sus observaciones"

HISTORIA DE USUARIO		
No: 14	**Iteración asignada:** 5	**Nombre:** Obtener el intervalo de confianza de una variable aleatoria a partir de sus observaciones.
Prioridad del negocio: Alta		**Nivel de complejidad:** Medio
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema devuelve un intervalo de confianza de una variable aleatoria a partir de sus observaciones y del valor de un estadígrafo.		
Observaciones: Los campos correspondientes a los valores de los parámetros no pueden estar vacíos.		

Anexo 14: Descripción de la HU "Guardar una lista de números aleatorios u observaciones aleatorias"

HISTORIA DE USUARIO		
No: 15	**Iteración asignada:** 1	**Nombre:** Guardar una lista de números aleatorios u observaciones aleatorias.
Prioridad del negocio: Alta		**Nivel de complejidad:** Bajo
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema será capaz de guardar una lista de números aleatorios u observaciones aleatorias.		
Observaciones:		

Anexo 15: Descripción de la HU “Cargar una lista de números aleatorios u observaciones aleatorias”

HISTORIA DE USUARIO		
No: 16	**Iteración asignada:** 2	**Nombre:** Cargar una lista de números aleatorios u observaciones aleatorias.
Prioridad del negocio: Alta		**Nivel de complejidad:** Bajo
Punto de estimación: 1		**Riesgo en el desarrollo:** Bajo
Descripción: El sistema será capaz de cargar una lista de números aleatorios u observaciones aleatorias.		
Observaciones:		

Anexo 16: Descripción de la tarjeta CRC “urls.py”

TARJETA CRC	
Clase: urls.py	
Responsabilidades:	**Colaboraciones**
Clase encargada de gestionar todas las peticiones que realicen los estudiantes, las cuales son ubicadas en el views.py en dependencia de la petición realiza.	Se relaciona con las clases: • views.py

Anexo 17: Descripción de la tarjeta CRC “models.py”

TARJETA CRC	
Clase: models.py	
Responsabilidades:	**Colaboraciones**
Clase encargada de validarlos los números aleatorios que el usuario carga al sistema.	Se relaciona con las clases: • views.py

Anexo 18: Descripción de la tarjeta CRC “validators.py”

TARJETA CRC	
Clase: validators.py	
Responsabilidades:	**Colaboraciones**

Clase encargada de validar si los parámetros introducidos para una sucesión de números aleatorios son de ciclo completo.	Se relaciona con las clases: • views.py

Anexo 19: Descripción de la tarjeta CRC “gna_mixto.html”

TARJETA CRC	
Clase: gna_mixto.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de ejecutar el método congruencial mixto.	

Anexo 20: Descripción de la tarjeta CRC “gna_aditivo.html”

TARJETA CRC	
Clase: gna_aditivo.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de ejecutar el método congruencial aditivo.	

Anexo 21: Descripción de la tarjeta CRC “gna_tausworthe.html”

TARJETA CRC	
Clase: gna_tausworthe.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de obtener números aleatorios utilizando generadores de Tausworthe.	

Anexo 22: Descripción de la tarjeta CRC “gna_multiplicativo.html”

TARJETA CRC	
Clase: gna_multiplicativo.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de ejecutar el método congruencial multiplicativo.	

Anexo 23: Descripción de la tarjeta CRC “tu_test_chi_cuadrado.html”

TARJETA CRC	
Clase: tu_test_chi_cuadrado.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de obtener el valor del estadígrafo X^2.	

Anexo 24: Descripción de la tarjeta CRC “tu_test_kolmogorov_smirnov.html”

TARJETA CRC	
Clase: tu_test_kolmogorov_smirnov.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de obtener el valor del estadígrafo D.	

Anexo 25: Descripción de la tarjeta CRC “ti_carreras.html”

TARJETA CRC	
Clase: ti_ carreras.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de obtener el valor del estadígrafo Z.	

Anexo 25: Descripción de la tarjeta CRC “ti_huecos.html”

TARJETA CRC	
Clase: ti_ huecos.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de obtener el valor del estadígrafo D.	

Anexo 26: Descripción de la tarjeta CRC “ti_autocorrelacion.html”

TARJETA CRC	
Clase: ti_ autocorrelacion.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos de la vista luego de obtener el valor del estadígrafo Z.	

Anexo 27: Descripción de la tarjeta CRC “goa_inversa.html”

TARJETA CRC	
Clase: goa_inversa.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar las observaciones obtenidas de la vista luego de ejecutar el método de la inversa.	

Anexo 28: Descripción de la tarjeta CRC “goa_normal.html”

TARJETA CRC	
Clase: goa_normal.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar las observaciones obtenidas de la vista luego de ejecutar el método de la normal.	

Anexo 29: Descripción de la tarjeta CRC “goa_uniforme.html”

TARJETA CRC	
Clase: goa_uniforme.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar las observaciones obtenidas de la vista luego de ejecutar una distribución uniforme.	

Anexo 30: Descripción de la tarjeta CRC “goa_composicion.html”

TARJETA CRC	
Clase: goa_composicion.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar las observaciones obtenidas de la vista luego de ejecutar el método de composición.	

Anexo 31: Descripción de la tarjeta CRC “goa_convolucion.html”

TARJETA CRC	
Clase: goa_ convolucion.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar las observaciones obtenidas de la vista luego de ejecutar el método de convolución.	

Anexo 31: Descripción de la tarjeta CRC “intervalo_confianza.html”

TARJETA CRC	
Clase: intervalo_confianza.html	
Responsabilidades:	**Colaboraciones**
Clase encargada de renderizar los resultados obtenidos después de determinar un intervalo de confianza.	

Anexo 32: Caso de prueba correspondiente a la HU "Generar números aleatorios utilizando el método congruencial mixto"

```
class gna_mixto_test(TestCase):
    def setUp(self):
        self.client = Client()

    def test_details(self):
        response = self.client .post(
            '/gna/mixto/',
            {
                'cifras': '2',
                'xo': '5',
                'a': '7',
                'c': '2',
                'm': '13',
                'k': '5',
            }
        )

        nums = [0.85, 0.08, 0.69, 0, 0.15]
        for i in range(len(response.context['nums'])):
            self.assertEqual(response.context['nums'][i]['r'], nums[i])

```

Anexo 33: Caso de prueba correspondiente a la HU "Generar números aleatorios utilizando el método congruencial aditivo"

```
class gna_aditivo_test(TestCase):
    def setUp(self):
        self.client = Client()

    def test_details(self):
        response = self.client .post(
            '/gna/aditivo/',
            {
                'cifras': '2',
                'numbers': '30,15,16,12,80',
                'i': '2',
                'j': '3',
                'm': '12',
                'k': '3',
            }
        )

        nums = [0.58,0.33,0.67]
        for i in range(len(response.context['rows'])):
            self.assertEqual(response.context['rows'][i][4], nums[i])

```

Anexo 34: Caso de prueba correspondiente a la HU "Generar números aleatorios utilizando el método congruencial multiplicativo"

```
class gna_multiplicativo_test(TestCase):
    def setUp(self):
        self.client = Client()

    def test_details(self):
        response = self.client .post(
            '/gna/multiplicativo/',
            {
                'cifras': '2',
                'xo': '8',
                'a': '7',
                'm': '13',
                'k': '3',
            }
        )

        nums = [0.31, 0.15, 0.08]
        for i in range(len(response.context['nums'])):
            self.assertEqual(response.context['nums'][i]['r'], nums[i])


```

Anexo 35: Caso de prueba correspondiente a la HU "Generar números aleatorios utilizando generadores de Tausworthe

```
class gna_tausworthe_test(TestCase):
    def setUp(self):
        self.client = Client()

    def test_details(self):
        response = self.client .post(
            '/gna/tausworthe/',
            {
                'secuencia': '01101',
                'r': '2',
                'q': '3',
            }
        )

        nums = [1, 1]
        for i in range(len(response.context['rows'])):
            self.assertEqual(response.context['rows'][i][4], nums[i])
```

"

Anexo 36: Caso de prueba correspondiente a la HU "Generar observaciones de variables aleatorias utilizando el método de la transformada inversa para la distribución exponencial"

```
184 class transformada_inversa_exponencial_test(TestCase):
185     def setUp(self):
186         self.client = Client()
187 
188     def test_details(self):
189         response = self.client .post(
190             '/goa/transformada_inversa/',
191             {
192                 'cifras': '2',
193                 'k': '1',
194                 'u': '3',
195                 'distribucion': 'exponencial',
196                 'numbers': '0.54',
197             }
198         )
199 
200         self.assertEqual(response.context['rows'][0][1], 1.85)
```

Anexo 37: Caso de prueba correspondiente a la HU "Generar observaciones de variables aleatorias utilizando el método de la transformada inversa para la distribución probabilística"

```
223 class transformada_inversa_probabilistica_test(TestCase):
224     def setUp(self):
225         self.client = Client()
226 
227     def test_details(self):
228         response = self.client .post(
229             '/goa/transformada_inversa/',
230             {
231                 'cifras': '2',
232                 'k': '3',
233                 'cantidad_intervalos': '3',
234                 'desde_0': '0',
235                 'hasta_0': '0.2',
236                 'decision_0': 'rojo',
237                 'desde_1': '0.2',
238                 'hasta_1': '0.5',
239                 'decision_1': 'verde',
240                 'desde_2': '0.5',
241                 'hasta_2': '1',
242                 'decision_2': 'azul',
243                 'distribucion': 'probabilistica',
244                 'numbers': '0.01,0.67,0.21',
245             }
246         )
247 
248         respuestas = ['rojo', 'azul', 'verde']
249         for i in range(len(response.context['rows'])):
250             self.assertEqual(response.context['rows'][i][1],
```

Anexo 38: Caso de prueba correspondiente a la HU "Generar observaciones de variables aleatorias utilizando el método de la transformada inversa para la distribución uniforme"

```
203 class transformada_inversa_uniforme_test(TestCase):
204     def setUp(self):
205         self.client = Client()
206 
207     def test_details(self):
208         response = self.client .post(
209             '/goa/transformada_inversa/',
210             {
211                 'cifras': '2',
212                 'k': '1',
213                 'a': '5',
214                 'b': '11',
215                 'distribucion': 'uniforme',
216                 'numbers': '0.35',
217             }
218         )
219 
220         self.assertEqual(response.context['rows'][0][1], 7.1)
221 
```

Anexo 39: Caso de prueba correspondiente a la HU "Generar observaciones de variables aleatorias utilizando el método de Box Müller"

```
253 class normal_box_muller_test(TestCase):
254     def setUp(self):
255         self.client = Client()
256 
257     def test_details(self):
258         response = self.client .post(
259             '/goa/normal/',
260             {
261                 'cifras': '2',
262                 'k': '1',
263                 'u': '4',
264                 'o': '3',
265                 'distribucion': 'normal',
266                 'numbers': '0.56,0.15',
267             }
268         )
269 
270         nums = [5.9]
271         for i in range(len(response.context['rows'])):
272             self.assertEqual(response.context['rows'][i][1], nums[i])
273 
```

Anexo 40: Caso de prueba correspondiente a la HU "Generar observaciones de variables aleatorias utilizando el teorema del límite central"

```
class normal_limite_test(TestCase):
    def setUp(self):
        self.client = Client()

    def test_details(self):
        response = self.client .post(
            '/goa/normal/',
            {
                'cifras': '2',
                'k': '1',
                'u': '1',
                'o': '3',
                'distribucion': 'limite',
                'numbers': '0.90,0.13,0.25,0.75,0.41,0.52,0.63,0.24,0.12,0.98,0.02,0.98',
            }
        )

        nums = [0.79]
        for i in range(len(response.context['rows'])):
            self.assertEqual(response.context['rows'][i][1], nums[i])

```

Anexo 41: Caso de prueba correspondiente a la HU "Generar observaciones de variables aleatorias utilizando el método de convolución"

```
class convolucion_test(TestCase):
    def setUp(self):
        self.client = Client()

    def test_details(self):
        response = self.client .post(
            '/goa/convolucion/',
            {
                'k': '2',
                'cifras': '2',
                'cantidad_distribuciones': '2',
                'distribucion_1': '1',
                'a_1': '2',
                'b_1': '5',
                'distribucion_2': '2',
                'u_2': '6',
                'numbers': '0.25,0.31,0.30,0.75',
            }
        )

        nums = [9.78, 4.63]
        for i in range(len(response.context['rows'])):
            self.assertEqual(float(response.context['rows'][i][1]), nums[i])
```

Anexo 42: Caso de prueba correspondiente a la HU "Generar observaciones de variables aleatorias utilizando el método de composición"

```
297 class composicion_test(TestCase):
298     def setUp(self):
299         self.client = Client()
300 
301     def test_details(self):
302         response = self.client .post(
303             '/goa/composicion/',
304             {
305                 'k': '1',
306                 'cantidad_intervalos': '2',
307                 'alias_0': 'X',
308                 'desde_0': '0',
309                 'hasta_0': '0.3',
310                 'lista_numeros_0': '2.6',
311                 'alias_1': 'Z',
312                 'desde_1': '0.3',
313                 'hasta_1': '1',
314                 'lista_numeros_1': '27.24',
315                 'numbers': '0.4',
316             }
317         )
318 
319         nums = [27.24]
320         for i in range(len(response.context['rows'])):
321             self.assertEqual(float(response.context['rows'][i][2]), nums[i])
322 
```

Anexo 43: Caso de prueba correspondiente a la HU "Validar la uniformidad de sucesiones de números aleatorios utilizando el test de Chi Cuadrado"

```
88  class chi_cuadrado_test(TestCase):
89      def setUp(self):
90          self.client = Client()
91  
92      def test_details(self):
93          response = self.client .post(
94              '/testu/chi_cuadrado/',
95              {
96                  'num_clases': '5',
97                  'cifras': '2',
98                  'numbers': '0.35,0.05,0.49,0.94,0.36,0.38,0.09,0.87,0.76,'
99                             '0.77,0.46,0.90,0.32,0.62,0.84,0.97,0.89,0.31,0.03,0.51',
100             }
101         )
102 
103         self.assertEqual(response.context['result'], 2)
104 
```

Anexo 44: Caso de prueba correspondiente a la HU "Validar la independencia en sucesiones de números aleatorios utilizando el test de las carreras por encima y por debajo de la media"

```
122 class carreras_test(TestCase):
123     def setUp(self):
124         self.client = Client()
125 
126     def test_details(self):
127         response = self.client .post(
128             '/testi/carreras/',
129             {
130                 'cifras': '4',
131                 'numbers': '0.02,0.17,0.21,0.54,0.86,0.06,0.49,0.52,0.75,'
132                            '0.29,0.09,0.72,0.07,0.55,0.91,0.45,0.20,0.43,'
133                            '0.08,0.66,0.69,0.85,0.01,0.11,0.83,0.84,0.95,'
134                            '0.72,0.13,0.94,0.61,0.35,0.54,0.80,0.68,0.79,'
135                            '0.62,0.41,0.92,0.50',
136             }
137         )
138 
139         self.assertEqual(response.context['n1'], 23)
140         self.assertEqual(response.context['n2'], 17)
141         self.assertEqual(response.context['Z'], -0.6723)
142 
```

Anexo 45: Caso de prueba correspondiente a la HU "Validar la independencia en sucesiones de números aleatorios utilizando el test de los huecos"

```
144 class huecos_test(TestCase):
145     def setUp(self):
146         self.client = Client()
147 
148     def test_details(self):
149         response = self.client .post(
150             '/testi/huecos/',
151             {
152                 'cifras': '4',
153                 'k': '4',
154                 'numbers': '0.33,0.04,0.96,0.87,0.91,0.06,0.10,0.40,0.56,0.67,'
155                            '0.31,0.99,0.26,0.48,0.60,0.97,0.71,0.54,0.66,0.19',
156             }
157         )
158 
159         self.assertEqual(response.context['D'], 0.1094)
160 
```

Anexo 46: Caso de prueba correspondiente a la HU "Validar la independencia en sucesiones de números aleatorios utilizando el test de autocorrelación"

```
162 class autocorrelacion_test(TestCase):
163     def setUp(self):
164         self.client = Client()
165 
166     def test_details(self):
167         response = self.client .post(
168             '/testi/autocorrelacion/',
169             {
170                 'cifras': '4',
171                 'i': '8',
172                 'm': '8',
173                 'numbers': '0.02,0.17,0.21,0.54,0.86,0.06,0.49,0.52,0.75,'
174                            '0.29,0.09,0.72,0.07,0.55,0.91,0.45,0.20,0.43,'
175                            '0.08,0.66,0.69,0.85,0.01,0.11,0.83,0.84,0.95,'
176                            '0.72,0.13,0.94,0.61,0.35,0.54,0.80,0.68,0.79,'
177                            '0.62,0.41,0.92,0.50',
178             }
179         )
180 
181         self.assertEqual(response.context['Z'], -0.8903)
```

Anexo 47: Caso de prueba correspondiente a la HU "Obtener el intervalo de confianza de una variable aleatoria a partir de sus observaciones"

```
324 class intervalo_confianza_test(TestCase):
325     def setUp(self):
326         self.client = Client()
327 
328     def test_details(self):
329         response = self.client .post(
330             '/intervalo_confianza/',
331             {
332                 'cifras': '5',
333                 'metodo': 'intervalo',
334                 'e': '2.2622',
335                 'numbers': '1.90,2.14,2.40,2.76,2.49,2.50,1.85,1.88,2.61,2.30',
336             }
337         )
338 
339         intervalo = '( 2.04992; 2.51608 )'
340         self.assertEqual(response.context['intervalo'], intervalo)
```

Anexo 48: Carta de aceptación del departamento de Ciencias Básicas

UCI
Universidad de las Ciencias Informáticas

AVAL SOBRE LA CALIDAD DE LA HERRAMIENTA SIMUCI

06 de junio de 2016
"Año 58 de la Revolución"

A quien pueda interesar:

En el departamento de Ciencias Básicas de la Facultad 3 de la UCI, se está apostando desde el curso escolar 2014-2015 por el diseño e implementación de un conjunto de soluciones bajo la concepción de un mismo marco de trabajo como apoyo a las asignaturas asociadas a dicho departamento. En un principio se comenzó con MATHNUM®, potente instrumento matemático que facilita la representación numérica y geométrica de las soluciones aproximadas según los métodos estudiados en la asignatura Matemática IV. La idea de incorporar herramientas de apoyo a dichas asignaturas viene dada principalmente por la poca motivación que sienten los estudiantes de la carrera por el estudio de las Ciencias Básicas. Además, la introducción de herramientas al proceso de enseñanza aprendizaje provee de nuevos mecanismos para que los estudiantes se nutran de los conocimientos necesarios y básicos de cada disciplina.

La solución que se ofrece en el trabajo de diploma titulado: **SIMUCI: Herramienta de apoyo a la asignatura Simulación** del autor: Joenis Rodríguez Fernández constituye un aporte de gran valor teórico-práctico para aplicar en las clases de la asignatura a la que corresponde y apoyar el proceso de enseñanza aprendizaje de la misma. Con el uso de esta herramienta se disminuye considerablemente el tiempo dedicado en las clases a la realización de operaciones matemáticas, posibilitando proceder al debate y al análisis de las soluciones que ofrecen métodos similares con iguales parámetros.

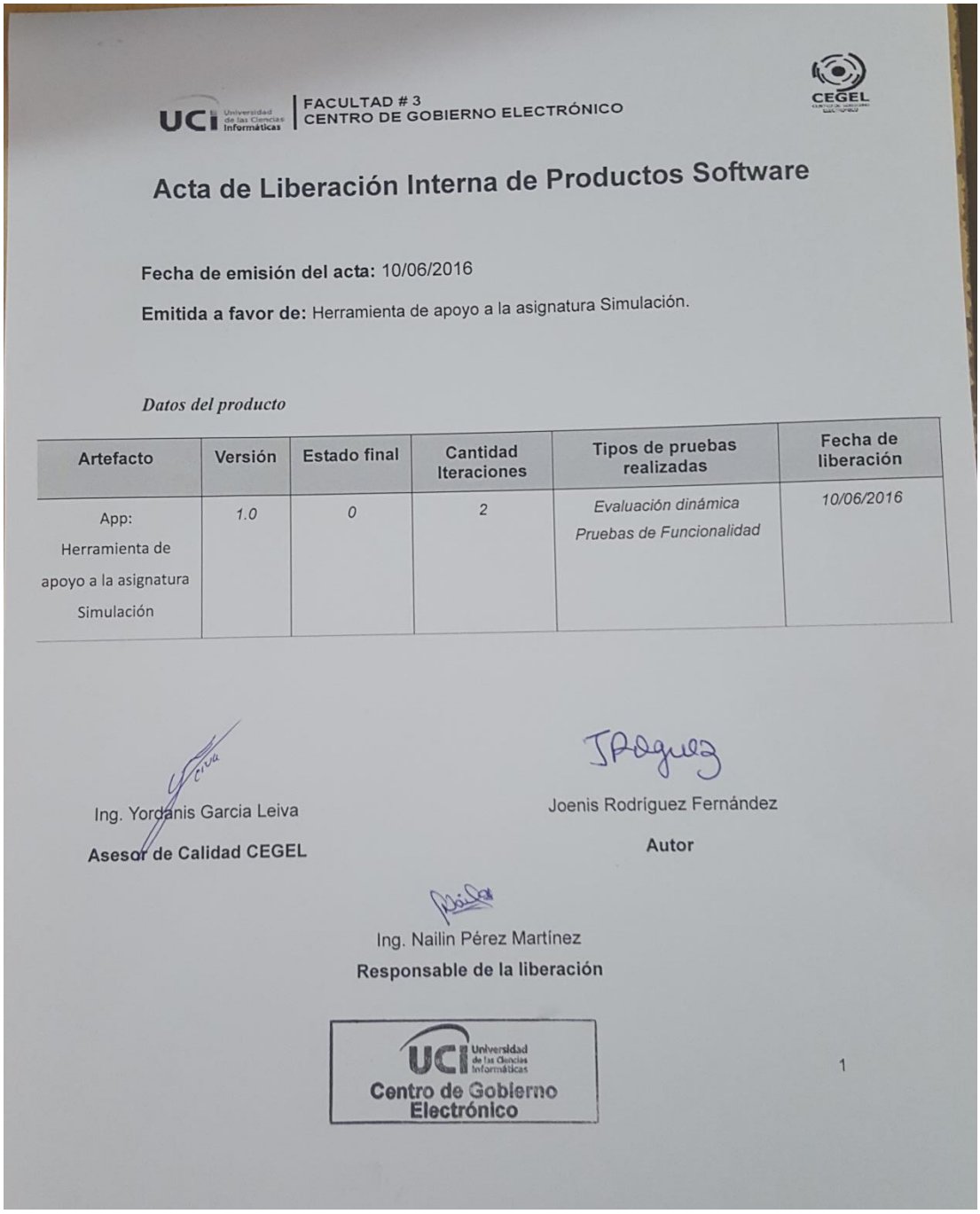
Se considera que la investigación es coherente, presenta una estructura acorde a un trabajo científico con buena ortografía y redacción de manera general. El mayor aporte consiste en que el investigador ofrece una solución que de incorporarse a las clases de la asignatura Simulación permitirá reforzar varios de los actuales procesos docentes que en ella se realizan. SIMUCI permite la presentación de los resultados parciales y métodos que se utilizan en la asignatura, favoreciendo a la comprensión de los mismos. La herramienta cumple con un extenso número de necesidades que han sido identificadas por el claustro de profesores.

Lo anterior permite validar la herramienta SIMUCI como un software con altas prestaciones tecnológicas, que cumple con las necesidades del departamento y puede utilizarse como apoyo a la asignatura Simulación.

MSc. Elizabeth Rodríguez Stiven
Profesora Auxiliar.
Jefe de Departamento de Ciencias Básicas
Facultad 3

Ing. Ariam Jorge Pedrosa González
Profesor de la asignatura Simulación
Facultad 3

Anexo 49: Carta de liberación interna de productos software



UCI Universidad de las Ciencias Informáticas | FACULTAD # 3
CENTRO DE GOBIERNO ELECTRÓNICO

CEGEL

Acta de Liberación Interna de Productos Software

Fecha de emisión del acta: 10/06/2016

Emitida a favor de: Herramienta de apoyo a la asignatura Simulación.

Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
App: Herramienta de apoyo a la asignatura Simulación	*1.0*	*0*	*2*	*Evaluación dinámica* *Pruebas de Funcionalidad*	*10/06/2016*

Ing. Yordanis Garcia Leiva
Asesor de Calidad CEGEL

Joenis Rodriguez Fernández
Autor

Ing. Nailin Pérez Martínez
Responsable de la liberación

UCI Universidad de las Ciencias Informáticas
Centro de Gobierno Electrónico

1

Anexo 50: Diseño de caso de prueba de aceptación para la HU: “Generar números aleatorios utilizando el método congruencial mixto”

CASO DE PRUEBA DE ACEPTACIÓN			
HISTORIA DE USUARIO		Generar números aleatorios utilizando el método congruencial mixto.	
DESCRIPCIÓN		El sistema genera una lista de números aleatorios utilizando el método congruencial mixto.	
ESCENARIO DE PRUEBA		**FLUJO DEL ESCENARIO**	**RESULTADOS ESPERADOS**
EP1	El usuario introduce parámetros de entrada válidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada correctamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una lista de números aleatorios.
EP2	El usuario introduce parámetros de entrada inválidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada incorrectamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una excepción detallando el error cometido.

Anexo 51: Diseño de caso de prueba de aceptación para la HU: “Generar números aleatorios utilizando el método congruencial aditivo”

CASO DE PRUEBA DE ACEPTACIÓN			
HISTORIA DE USUARIO		Generar números aleatorios utilizando el método congruencial aditivo.	
DESCRIPCIÓN		El sistema genera una lista de números aleatorios utilizando el método congruencial aditivo.	
ESCENARIO DE PRUEBA		**FLUJO DEL ESCENARIO**	**RESULTADOS ESPERADOS**
EP1	El usuario introduce parámetros de entrada válidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada correctamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una lista de números aleatorios.
EP2	El usuario introduce parámetros de entrada inválidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada incorrectamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una excepción detallando el error cometido.

Anexo 52: Diseño de caso de prueba de aceptación para la HU: “Generar números aleatorios utilizando el método congruencial multiplicativo”

CASO DE PRUEBA DE ACEPTACIÓN			
HISTORIA DE USUARIO		Generar números aleatorios utilizando el método congruencial multiplicativo.	
DESCRIPCIÓN		El sistema genera una lista de números aleatorios utilizando el método congruencial multiplicativo.	
ESCENARIO DE PRUEBA		**FLUJO DEL ESCENARIO**	**RESULTADOS ESPERADOS**
EP1	El usuario introduce parámetros de entrada válidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada correctamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una lista de números aleatorios.
EP2	El usuario introduce parámetros de entrada inválidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada incorrectamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una excepción detallando el error cometido.

Anexo 53: Diseño de caso de prueba de aceptación para la HU: “Generar números aleatorios utilizando generadores de Tausworthe”

CASO DE PRUEBA DE ACEPTACIÓN			
HISTORIA DE USUARIO		Generar números aleatorios utilizando generadores de Tausworthe.	
DESCRIPCIÓN		El sistema genera una lista de números aleatorios utilizando generadores de Tausworthe.	
ESCENARIO DE PRUEBA		**FLUJO DEL ESCENARIO**	**RESULTADOS ESPERADOS**
EP1	El usuario introduce parámetros de entrada	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada correctamente.	El sistema devuelve una lista de números aleatorios.

	válidos para realizar los cálculos	3. Se presiona el botón Aceptar.	
EP2	El usuario introduce parámetros de entrada inválidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada incorrectamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una excepción detallando el error cometido.

Anexo 54: Diseño de caso de prueba de aceptación para la HU: "Validar la uniformidad de sucesiones de números aleatorios utilizando el test de Chi Cuadrado"

CASO DE PRUEBA DE ACEPTACIÓN			
HISTORIA DE USUARIO		Validar la uniformidad de sucesiones de números aleatorios utilizando el test de Chi Cuadrado.	
DESCRIPCIÓN		Utilizar el Test de Chi Cuadrado para validar que una sucesión de números aleatorios se encuentre distribuida uniformemente en el intervalo de 0 a 1.	
CONDICIONES DE EJECUCIÓN		Haber generado previamente un listado de números aleatorios.	
ESCENARIO DE PRUEBA		**FLUJO DEL ESCENARIO**	**RESULTADOS ESPERADOS**
EP1	El usuario introduce parámetros de entrada válidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada correctamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una tabla auxiliar donde se calcula el de valor de X^2.
EP2	El usuario introduce parámetros de entrada inválidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada incorrectamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una excepción detallando el error cometido.

Anexo 63: Diseño de caso de prueba de aceptación para la HU: "Validar la independencia de sucesiones de números aleatorios utilizando el test de los huecos"

CASO DE PRUEBA DE ACEPTACIÓN			
HISTORIA DE USUARIO		Validar la independencia en sucesiones de números aleatorios utilizando el test de los huecos.	
DESCRIPCIÓN		Utilizar el test de los huecos para validar que una sucesión de números aleatorios son independientes	
CONDICIONES DE EJECUCIÓN		Haber generado previamente un listado de números aleatorios.	
ESCENARIO DE PRUEBA		**FLUJO DEL ESCENARIO**	**RESULTADOS ESPERADOS**
EP1	El usuario introduce parámetros de entrada válidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada correctamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una tabla auxiliar donde se calcula el valor de D.
EP2	El usuario introduce parámetros de entrada inválidos para realizar los cálculos	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada incorrectamente. 3. Se presiona el botón Aceptar.	El sistema devuelve una excepción detallando el error cometido.

Anexo 64: Diseño de caso de prueba de aceptación para la HU: "Validar la independencia en sucesiones de números aleatorios utilizando el test de las carreras por encima y por debajo de la media"

CASO DE PRUEBA DE ACEPTACIÓN			
HISTORIA DE USUARIO		Validar la independencia en sucesiones de números aleatorios utilizando el test de las carreras por encima y por debajo de la media.	
DESCRIPCIÓN		Utilizar el test de los huecos para validar que una sucesión de números aleatorios son independientes	
CONDICIONES DE EJECUCIÓN		Haber generado previamente un listado de números aleatorios.	
ESCENARIO DE PRUEBA		**FLUJO DEL ESCENARIO**	**RESULTADOS ESPERADOS**
EP1	El usuario introduce parámetros de entrada	1. Se presiona el botón Cargar. 2. Se introducen los parámetros de entrada correctamente.	El sistema devuelve una tabla auxiliar donde se calcula el valor de Z.

	válidos para realizar los cálculos	3. Se presiona el botón Aceptar.	
EP2	El usuario introduce parámetros de entrada inválidos para realizar los cálculos	4. Se presiona el botón Cargar. 5. Se introducen los parámetros de entrada incorrectamente. 6. Se presiona el botón Aceptar.	El sistema devuelve una excepción detallando el error cometido.