Universidad de las Ciencias Informáticas

Facultad 3

ANÁLISIS Y DISEÑO DEL SIMULADOR DE INTERBLOQUEOS DEL LABORATORIO VIRTUAL DE SISTEMAS OPERATIVOS.

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

Autor: Leonardo Carbonell Leyva

Tutor: Ing. Carlos Y. Hidalgo García.

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

Albert Einstein

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Leonardo Carbonell Leyva _____ Carlos Y. Hidalgo García _____

Firma del Autor Firma del Tutor

Dedicatoria

Dedico este trabajo especialmente a mis padres que tanto han hecho y han dado para ver mis sueños hechos realidad. A mima por estar siempre allí cuidándome y guiándome. A mi abuelo Yoel por forjar mis principios revolucionarios y por enseñarme a no rendirme jamás. A la memoria de mi abuelo Carlos Carbonell que no se encuentra físicamente, pero su presencia sigue entre nosotros. A mi abuelita Olga que siempre se preocupa por mí. A toda la familia que de una forma u otra me han apoyado en todo y que gracias a eso he logrado este triunfo, que más que mío es de todos.

Agradecimientos

A toda mi familia por estar al tanto de mis estudios por todos estos años.

En especial a la mejor madre del mundo por sus preocupaciones constantes y sus consejos oportunos.

A mi papá por estar siempre allí cuando me hizo falta y por darme su ejemplo de persona excepcional.

A Mima, abuela Olga, abuelo Yoel, y abuelo Carlos por estar siempre al tanto de mis estudios.

A mi hermano Raulito por mostrarme el camino del profesionalismo y del estudio abnegado.

A mi Cielo, amor y futura esposa que tanta fuerza y esperanza me brindó a lo largo de la tesis y que por fin vamos a poder estar juntos por siempre.

A Ismaris, Pedro y toda la familia de Alquízar que tanto me ayudaron incondicionalmente.

A Oscar, JC, Yoisdel, Eber, Wilfredo, Lumianys y el Flaco, mis amigos y colegas de siempre, por toda la ayuda brindada a lo largo de la carrera.

A mis amigos y compañeros de la universidad, los que están y los que por una razón u otra ya no están en especial a Idanis, Beilen, Laritza y Yordan que me tendieron su brazo en muchas ocasiones.

A mis profesores por enseñarme a ser un profesional a lo largo de seis años de estudios en la UCI.

A todos aquellos que de una forma u otra ayudaron con la realización de esta tesis.

Gracias.

Resumen

Teniendo en cuenta la necesidad del aumento de la calidad en la educación cubana, en la Universidad de las Ciencias Informáticas (UCI) se ha implementado un modelo de integración donde los procesos de Formación – Producción e Investigación se funden en uno solo para aumentar la calidad del egresado, en el que se destaca la utilización de recursos didácticos y el perfeccionamiento del plan de estudios.

Como objeto de la investigación se tomó la asignatura de Sistemas Operativos, para la que se está desarrollando un laboratorio virtual con el objetivo de contribuir a la incorporación de la asignatura al modelo de formación y dentro del que se encuentra un subsistema de Simuladores, para el que se realiza la modelación del simulador de bloqueos.

En el desarrollo de la propuesta se utilizó la metodología de desarrollo RUP, como lenguaje de modelado UML 2.1 y Visual Paradigm 8.0 como herramienta CASE para la obtención de los modelos, lo que permitió obtener los artefactos necesarios para su posterior implementación.

Para la validación de los requisitos se aplicó la técnica de prototipos, se aplicaron métricas de calidad de la especificación de requisitos y de casos de usos además de métricas para la validación del diseño, obteniéndose en todos los casos resultados satisfactorios.

Palabras Claves: Sistemas Operativos, simuladores, laboratorio virtual, interbloqueos.

Índice de contenido

Índice de ilustraciones

Índice de Tablas

Introducción

A través de la historia el hombre ha necesitado tratar y transmitir información de forma continua, fiable y segura. Para eso creó una serie de mecanismos y técnicas tales como las señales de humo, la escritura, mensajes manuscritos, destellos con espejos. Con este fin surge la Informática, el término Informática se creó en Francia en el año 1962 bajo la denominación INFORMATIQUE y procede de la contracción de las palabras INFORmation y autoMATIQUE. En otras palabras la informática es la ciencia encargada de estudiar y desarrollar máquinas y métodos para el tratamiento automatizado de la información. (Levene, y otros, 1998)

En el año 2002 el comandante en jefe Fidel Castro crea la Universidad de las Ciencias Informática (UCI), en el marco de la batalla de ideas con el objetivo de contribuir a la informatización de la sociedad. Un proyecto que tomaría un gran auge debido a la alta preparación profesional de los egresados que aportaría a la sociedad, así como los avances investigativos en la ciencia de la informática y las comunicaciones.

La UCI está llamada a ser una universidad de nuevo tipo debido a los programas de estudio que desarrolla así como la integración docencia-producción-investigación con la que se preparan sus estudiantes. La UCI aspira a ser el motor impulsor del desarrollo de software en Cuba y para esto se ha visto implicada en un aumento de sus compromisos profesionales con empresas cubanas y del exterior del país.

A partir de una creciente vinculación de profesores y estudiantes a las labores productivas, se concibió un nuevo modelo de formación siguiendo los siguientes principios: centrado en el aprendizaje, establecimiento de un ciclo de formación básico y otro profesional en los cuales tienen un uso protagónico los entornos virtuales de aprendizaje y la incorporación de los estudiantes, a partir del segundo semestre de 3er año, a proyectos de desarrollo donde tienen la oportunidad de adquirir habilidades desde el punto de vista profesional en dependencia de los objetivos a alcanzar en cada uno de los años (UCI, 2012).

El uso del Entorno Virtual de Aprendizaje (EVA) en la UCI para el Proceso de Enseñanza - Aprendizaje (PEA) es creciente, en el que se destaca la utilización de recursos didácticos. En tal sentido es importante destacar el rol que desempeñan los laboratorios virtuales, siendo una significativa herramienta de apoyo al trabajo docente, tanto para el profesor como para el estudiante.

Como parte de la disciplina de Sistemas Digitales se imparte la asignatura de Sistemas Operativos en tercer año de la carrera. Esta está dividida en tres partes fundamentales: procesos, memoria y entrada/salida de información. La asignatura en su totalidad tiene poca dependencia con las tecnologías de la informática y las comunicaciones (TIC) lo cual es una contradicción ya que su estudio se basa específicamente en las TIC. Además el grueso de la enseñanza se centra en las clases presenciales lo cual entra en conflicto con el nuevo modelo de formación donde la semipresencialidad y el uso de las TIC por los estudiantes, son imprescindibles para desarrollar sus habilidades investigativas y de aprendizaje, donde ellos mismos son los protagonistas de su enseñanza - aprendizaje.

De las tres partes en las que está dividida la asignatura Sistemas Operativos, el tema dedicado a procesos es uno de los más complicados para el estudiante dado el grado de análisis que requiere y donde las TIC pueden jugar un papel fundamental como herramienta para el apoyo al aprendizaje de los contenidos correspondientes a ese tema. Específicamente en los contenidos referentes a interbloqueos se ha detectado que le resulta difícil al estudiante lograr la comprensión de los conceptos y técnicas a los cuales se hacen referencia, así como los mecanismos para detección, prevención y evasión del interbloqueo.

Los medios didácticos con los que se apoya al Proceso de Enseñanza - Aprendizaje en la UCI carecen de efectividad a la hora de brindar a los estudiantes una mayor comprensión y entendimiento en el tema de bloqueo de procesos en la asignatura de Sistemas Operativos. Lo anterior puede comprobarse en el informe semestral de la asignatura de Sistemas Operativos, del cual se han extraído algunas de las deficiencias (Roque & otros, 2012):

- La falta de motivación en los estudiantes por lo difícil de algunos de los temas, unido a la inexperiencia de algunos profesores provoca que los estudiantes olviden fácilmente el contenido una vez pasado el mismo, además de que el autoestudio les resulte muy complejo al contar solamente con los documentos de clases.
- La carencia de materiales didácticos en la asignatura conlleva a que los estudiantes no cuenten con los medios necesarios para aumentar la comprensión de los contenidos vistos en el aula así como cuenten con una forma de poner en práctica los contenidos teóricos que se imparten por parte de los profesores.

- En el trabajo de control parcial número dos, correspondiente al curso 2012-2013 donde se incluyeron temas referentes a procesos e interbloqueos, el porciento de estudiantes desaprobados fue por encima del 20%, un total de 55 estudiantes exactamente un 30,6 %.
- La mayoría de los estudiantes presentaron dificultades en las preguntas 2 y 4 del examen, correspondiente a los temas Mecanismos de comunicación entre procesos e Interbloqueo, respectivamente. Los porcientos de aprobados en estas preguntas fueron: pregunta 2 un 27,7% y en la pregunta 4 un 17,9 %. Se suma a esto, los resultados del examen final de la asignatura en el cual desaprobaron un total de 27 estudiantes lo que representa un 15,6% de los que se presentaron al examen.

A partir de un estudio previo realizado en cursos anteriores por parte de algunos profesores de la asignatura se decidió, y apoyado por los resultados de la asignatura en el 1er semestre de este curso, el desarrollo de un laboratorio virtual para la asignatura, como herramienta para apoyar el PEA de la asignatura, dándole la posibilidad al estudiante de contar con un entorno para poner en práctica los conocimientos adquiridos en clases y al mismo tiempo darle la posibilidad al profesor de contar con una herramienta para darle un seguimiento a cada estudiante y poder realizar acciones de atención individual y asesoría a aquellos estudiantes que cuyo aprendizaje no esté al mismo nivel que el resto.

Este laboratorio virtual, en su concepción, cuenta con 4 subsistemas y 1 módulo donde se agrupan todas las funcionalidades con las que contará esta herramienta de trabajo, destacándose el subsistema de Ejercicios por ser sobre el que el estudiante dirigirá una gran parte de sus actividades dentro del laboratorio.

Como parte de este subsistema se incluye el subsistema de Simuladores, los que se desarrollarán y se irán poniendo a disposición del estudiante para que, como medios didácticos, sirvan de apoyo al estudiante en virtud de consolidar los conocimientos respecto a la solución práctica de los ejercicios propuestos por el profesor tanto en el aula como para el estudio independiente. En el módulo correspondiente no se encuentra incluido un simulador que permita brindarle al estudiante la forma de resolución de los ejercicios respecto al tema de interbloqueos por lo que aún, el PEA de este tema por parte de los estudiantes, se sustenta solamente en documentos de clases y bibliografías complementarias recomendadas por el profesor trayendo como consecuencia que, entre otras razones asociadas, se hayan obtenido resultados históricos similares a los del pasado semestre del curso 2011-2012 en la asignatura, lo cual refleja una mala calidad en el PEA por parte de los estudiantes.

Tomando como referencia lo anteriormente expuesto tenemos como problema a resolver: ¿Cómo modelar el tema referente a interbloqueo de manera que facilite la posterior implementación de un simulador para el subsistema de simuladores del laboratorio virtual de Sistemas Operativos?

De aquí que el objetivo general sea: realizar el análisis y el diseño del simulador de interbloqueo para el Laboratorio Virtual de Sistemas Operativos y cuyo campo de acción es: análisis y diseño de simuladores de Interbloqueo para la asignatura de Sistemas Operativos.

Se tiene la siguiente idea a defender: con la obtención de los artefactos correspondientes al análisis y el diseño del simulador de Interbloqueos para el subsistema de simuladores del Laboratorio Virtual de Sistemas Operativos, se podrá proceder a la posterior implementación del mismo.

Para dar cumplimiento al objetivo general se tienen los siguientes objetivos específicos:

1. Elaborar el marco teórico de la investigación.
2. Obtener algunos de los artefactos generados en los flujos de trabajo de Modelo de negocio, Requerimientos y Análisis y Diseño.
3. Validar la solución propuesta mediante el uso de métricas de validación para cada artefacto específico.

Para desarrollar los objetivos específicos se proponen las siguientes tareas de la investigación:

1. Realización de un estudio que refleje el estado del arte respecto al desarrollo de simuladores así como de las herramientas, metodologías y técnicas a utilizar en la investigación.
2. Obtención del modelo de dominio inicial.
3. Obtención de la especificación de requisitos funcionales y no funcionales.
4. Obtención de los artefactos generados en la fase de análisis.
5. Especificación de la arquitectura a utilizar.
6. Obtención de los artefactos generados en la fase de diseño.
7. Realización de la validación de los artefactos obtenidos en las fases de análisis y diseño.

Métodos de la investigación

Métodos teóricos:

- Análisis Histórico – Lógico: Este método permite profundizar en los antecedentes de la utilización de las TIC en los procesos de enseñanza – aprendizaje y sus tendencias actuales en la asignatura de Sistemas Operativos.

- Analítico - Sintético: Este método se utiliza en el estudio de los informes con respecto a la estructura de la asignatura según lo establecido en el modelo de formación de la UCI y análisis de los mismos con vista conocer su funcionamiento para realizar una correcta toma de decisiones.

Métodos empíricos:

- Entrevista: Este método permite una rápida recopilación de la información y está dirigida a los directivos, profesores y alumnos que interactúan con las TIC en el proceso de enseñanza – aprendizaje en la asignatura de Sistemas Operativos en la facultad 3 de la UCI.

Capítulo I: Fundamentación Teórica

El presente capítulo contiene una breve descripción de los elementos más importantes a tener en cuenta en la fundamentación teórica. Entre los conceptos que se manejan los más importantes son laboratorios virtuales, simulación por computadora e interbloqueos de procesos. También se analiza la metodología de desarrollo de software a utilizar, lenguaje de modelado, herramientas y patrones de casos de uso con el fin de seleccionar las más adecuadas para el desarrollo del sistema que dará solución a la problemática existente.

1.1 Interbloqueos de procesos

Un interbloqueo se define como el bloqueo permanente de un conjunto de procesos que compiten por los recursos del sistema o bien se comunican unos con otros. A diferencia de otros problemas de la gestión concurrente de procesos, para el caso general no existe una solución eficiente (Stallings, 1997).

Para que ocurra un interbloqueo deben darse tres condiciones (exclusión mutua, retención y espera y no apropiación), estas condiciones son necesarias pero no suficientes. Se necesita una cuarta condición (circulo vicioso de espera) que es una consecuencia potencial de la ocurrencia de las tres primera, o sea producto de la ocurrencia de las tres primeras condiciones puede ocurrir una cuarta, de suceder esto se dan las cuatro condiciones suficientes y necesarias para que ocurra el interbloqueo. Estas cuatro condiciones son llamadas las condiciones de Coffman.

1.1.1 Condiciones del interbloqueo o condiciones de Coffman

- Exclusión mutua: Sólo un proceso puede usar un recurso simultáneamente.
- Retención y espera: Un proceso puede retener unos recursos asignados mientras espera que se le asignen otros.
- No apropiación: Ningún proceso puede ser forzado a abandonar un recurso que retenga.
- Círculo vicioso de espera: Existe una cadena cerrada de procesos, cada uno de los cuales retiene, al menos, un recurso que necesita el siguiente proceso de la cadena (Stallings, 1997).

1.1.2 Prevención, detección y predicción del interbloqueo

Según (Stallings, 1997) existen tres estrategias para enfrentar un interbloqueo:

- Prevención
- Detección y recuperación
- Predicción

La estrategia de prevención del interbloqueo consiste, a grandes rasgos, en diseñar un sistema de manera que esté excluida, a priori, la posibilidad de interbloqueo. Los métodos para prevenir el interbloqueo son de dos tipos. Los métodos indirectos consisten en impedir la aparición de alguna de las tres condiciones necesarias, antes mencionadas (condiciones 1 a 3). Los métodos directos consisten en evitar la aparición del círculo vicioso de espera (condición 4) (Stallings, 1997).

Las estrategias de prevención del interbloqueo solucionan el interbloqueo limitando el acceso a sus recursos e imponiendo restricciones a los procesos.

Por su lado las estrategias de detección del interbloqueo no limitan el acceso a los recursos ni restringen las acciones de los procesos. Con detección del interbloqueo, se concederán los recursos que los procesos necesiten siempre que sea posible. Periódicamente, el sistema operativo ejecuta un algoritmo que permite detectar la condición de círculo vicioso de espera. Puede emplearse cualquier algoritmo de detección de ciclos en grafos dirigidos (Stallings, 1997).

Una vez detectado el interbloqueo, hace falta alguna estrategia de recuperación. Las técnicas siguientes son posibles enfoques, enumeradas en orden creciente de sofisticación (Stallings, 1997):

- Abandonar todos los procesos bloqueados.
- Retroceder cada proceso interbloqueado hasta algún punto de control definido previamente y volver a ejecutar todos los procesos.
- Abandonar sucesivamente los procesos bloqueados hasta que deje de haber interbloqueo.
- Apropiarse de recursos sucesivamente hasta que deje de haber interbloqueo.

En la predicción del interbloqueo, por otro lado, se pueden alcanzar las tres condiciones necesarias, pero se realizan elecciones acertadas para asegurar que nunca se llega al punto de interbloqueo. La

predicción, por tanto, permite más concurrencia que la prevención. Con predicción del interbloqueo, se decide dinámicamente si la petición actual de asignación de un recurso podría, de concederse, llevar potencialmente a un interbloqueo. La predicción del interbloqueo necesita, por tanto, conocer las peticiones futuras de recursos (Stallings, 1997).

En la investigación se hará énfasis en la estrategia de detección y recuperación del interbloqueo y como métodos de solución se utilizarán los grafos de asignación de recursos y las matrices de asignación de recursos y como algoritmo se utilizará el algoritmo del banquero propuesto por Dijkstra.

Para los Sistemas Operativos modernos, un recurso es cualquier cosa (archivo, segmento de memoria, procesador, variable, interrupción, etc.) que solo pueda ser utilizada por un proceso o hilo a la vez, a pesar que pueden existir varias instancias de un mismo recurso como por ejemplo dos tarjetas de video o tres discos duros.

La concurrencia de procesos, además de posibilitar problemas de sincronización, también trae consigo problemas de interbloqueo. El interbloqueo siempre implica a más de un proceso o hilo y básicamente ocurre cuando cada proceso bloqueado está esperando por un recurso o señal de otro de los bloqueados. Como ninguno se puede ejecutar, la espera será eterna. He ahí que en inglés se denomine deadlock (abrazo fatal).

Es posible modelar gráficamente situaciones potenciales de bloqueo. Para ello se utilizan grafos de asignación de recursos.

Para manejar situaciones potenciales de bloqueo, resulta conveniente modelar las asignaciones de recursos en forma de vectores o matrices en lugar de grafos. De modo que se van a obtener:

1. Un vector de recursos existentes. Cada elemento del vector es la cantidad máxima de instancias de cada recurso en el sistema.
2. Un vector de recursos disponibles. Cada elemento del vector es la cantidad de instancias de cada recurso disponibles en un momento dado.
3. Una matriz de recursos asignados. Cada elemento de la matriz es la cantidad de instancias de cada recurso (en las columnas) asignadas a cada proceso (filas).
4. Una matriz de recursos solicitados (o demanda). Cada elemento de la matriz es la cantidad de instancias de cada recurso (en las columnas) solicitadas por cada proceso (filas).

En esencia, un bloqueo se detecta cuando no es posible satisfacer ninguna de las solicitudes de recursos.

1.2 Laboratorios virtuales (LV)

Según el informe de la reunión de expertos sobre LV, los LV son un espacio electrónico de trabajo concebido para la colaboración y la experimentación a distancia con el objeto de investigar o realizar actividades creativas, y elaborar y difundir resultados mediante tecnologías difundidas de información y comunicación (Vary, 2000).

Por otro lado tenemos que un LV es un sistema computacional que pretende aproximar el ambiente de un laboratorio tradicional (LT) (Rosado, y otros, 2009).

1.2.1 Ventajas e inconvenientes de utilizar LV

Algunas de las ventajas más importantes de los LV son (Rosado, y otros, 2009):

- Acerca y facilita a un mayor número de alumnos la realización de experiencias, aunque alumno y laboratorio no coincidan en el espacio. El estudiante accede a los equipos del laboratorio a través del navegador, pudiendo experimentar sin riesgo alguno, y además, se flexibiliza el horario de prácticas y evita la saturación por el solapamiento con otras asignaturas.
- Reducen el coste del montaje y mantenimiento de los LT, siendo una alternativa barata y eficiente, donde el estudiante simula los fenómenos a estudiar como si los observase en el LT.
- Es una herramienta de auto aprendizaje, donde el alumno altera las variables de entrada, configura nuevos experimentos, aprende el manejo de instrumentos, personaliza el experimento, etc. La simulación en el LV, permite obtener una visión más intuitiva de aquellos fenómenos que en su realización manual no aportan suficiente claridad gráfica. El uso de LV da lugar a cambios fundamentales en el proceso habitual de enseñanza, en el que se suele comenzar por el modelo matemático. La simulación interactiva de forma aislada posee poco valor didáctico, esta debe ser embebida dentro de un conjunto de elementos multimedia que guíen al alumno eficazmente en el proceso de aprendizaje. Se trata de utilizar la capacidad de procesamiento y cálculo del ordenador, incrementando la diversidad didáctica, como complemento eficaz de las metodologías más convencionales.

- Los estudiantes aprenden mediante prueba y error, sin miedo a sufrir o provocar un accidente, sin avergonzarse de realizar varias veces la misma práctica, ya que pueden repetirlas sin límite; sin temor a dañar alguna herramienta o equipo. Pueden asistir al laboratorio cuando ellos quieran, y elegir las áreas del laboratorio más significativas para realizar prácticas sobre su trabajo.

- En internet encontramos multitud de simulaciones de procesos físicos (en forma de applets de Java y/o Flash). Con estos objetos dinámicos, el docente puede preparar actividades de aprendizaje que los alumnos han de ejecutar, contestando al mismo tiempo las cuestiones que se les plantean.

No todo son ventajas en los LV, también existen inconvenientes. A continuación mostramos los más destacados (Rosado, y otros, 2009):

- El LV no puede sustituir la experiencia práctica altamente enriquecedora del LT. Ha de ser una herramienta complementaria para formar a la persona y obtener un mayor rendimiento.

- En el LV se corre el riesgo de que el alumno se comporte como un mero espectador. Es importante que las actividades en el LV, vengan acompañadas de un guión que explique el concepto a estudiar, así como las ecuaciones del modelo utilizado. Es necesario que el estudiante realice una actividad ordenada y progresiva, conducente a alcanzar objetivos básicos concretos.

- El alumno no utiliza elementos reales en el LV, lo que provoca una pérdida parcial de la visión de la realidad. Además, no siempre se dispone de la simulación adecuada para el tema que el profesor desea trabajar. En internet existe demasiada información, a veces inútil. Para que sea útil en el PEA, hemos de seleccionar los contenidos relevantes para nuestros alumnos. Son pocas las experiencias realizadas con LV en los centros educativos, donde aún impera el uso de recursos tradicionales, tanto en la exposición de conocimientos en el aula como en el laboratorio.

Por lo visto anteriormente podemos afirmar que los LV son espacios de trabajo apoyados en las TIC a los cuales se pueden acceder o no mediante la web y que pretenden simular a los LT pero nunca van a ser lo mismo. Los LV son de suma importancia en la UCI dado el nuevo modelo de enseñanza-aprendizaje, en los mismos podemos modelar procesos de diferentes asignaturas como lo pueden ser Física, SO, Matemática, entre otras y lograr que el estudiante visualice y comprenda de una forma más

fácil y didáctica lo que se les imparte en las clases. Con el uso de los LV se pretende mejorar la calidad en el proceso de aprendizaje sin caer en los inconvenientes mencionados anteriormente y aumentar de esta forma la preparación del estudiante.

1.3 Simulación por computadoras

1.3.1 ¿Qué es la simulación?

La esencia de la simulación consiste en establecer una equivalencia entre dos sistemas, cada uno de los cuales puede existir en realidad o ser abstracto. Si el primero resulta más sencillo para la investigación que el segundo, es posible juzgar sobre las propiedades del segundo sistema al observar el comportamiento del primero. En este caso el sistema empleado para la investigación se denomina modelo (Rodríguez Chávez, y otros, 2009).

Según R.E. Shannon: "La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias (dentro de los límites impuestos por un cierto criterio, o un conjunto de ellos) para el funcionamiento del sistema" (Rodríguez Chávez, y otros, 2009).

1.3.2 Simulación por computadoras

Las simulaciones por computadoras son programas que sostienen modelos de sistemas reales. El comportamiento de estos sistemas se expresa mediante cambios en las variables que lo describen. En caso que no sea posible representarlos todos, se selecciona una representación de los principales estados del sistema real (Rodríguez Chávez, y otros, 2009).

La Simulación es la imitación del funcionamiento de un sistema real durante un intervalo de tiempo. Esta simulación puede realizarse ya sea de forma manual o computacional. La simulación se basa en un modelo de la realidad que cuenta una historia y al observar el comportamiento de esta, nos permite obtener conocimiento acerca del sistema real. El comportamiento de la simulación está determinado por el modelo de simulación o conjunto de supuestos concernientes al sistema real, estos supuestos se expresan a través de relaciones lógicas y matemáticas entre las entidades (García, 2011).

1.3.3 Simulación en la enseñanza

Una simulación educativa es una poderosa técnica que enseña algunos aspectos del mundo mediante su imitación o réplica. Está basada en un modelo de un sistema o fenómeno del mundo real en el que se han simplificado u omitido algunos elementos para facilitar el aprendizaje.

Las simulaciones permiten colocar al alumno en situaciones de aprendizaje que, por restricciones económicas o físicas, son difíciles de obtener en una experiencia de laboratorio tradicional (Rodríguez Chávez, y otros, 2009).

Durante el PEA, los diversos tipos de simulación disponibles pueden utilizarse no sólo para el mejoramiento de las técnicas de diagnóstico, tratamiento y de resolución de problemas, sino también para mejorar las facultades psicomotoras y de relaciones humanas, donde en ocasiones pueden ser más eficaces que muchos métodos tradicionales, todo lo cual está en dependencia fundamentalmente de la fidelidad de la simulación (Salas Perea, y otros, 1995)

1.3.4 Tipos de simuladores

Simuladores de administración de empresas:

LABSAG es un laboratorio virtual de simuladores de administración y gerencia. El mismo cuenta con tres simuladores de gerencia general integral y seis simuladores con escenarios especializados funcionalmente. LABSAG es producida y comercializada por Michelsen Consulting Ltd. Actualmente se encuentra comercializándose la versión 5.0, LABSAG es un simulador web escrito en asp y visual basic con su servidor en internet. La dirección URL para acceder al LABSAG es la siguiente http://www.gerentevirtual.com/es/.

Este simulador no nos sirve para dar solución a la problemática existente pues es un simulador propietario por lo que hay que pagar para su utilización, además para acceder a su base de datos hay que conectarse a su servidor web en internet. Las PC instaladas con el simulador deben estar configuradas para tener una conexión a internet de modo que el intercambio de archivos se realice sin pasar por dispositivos de interconexión.

Simulador de conducción:

Simescar es un simulador de conducción desarrollado por la firma Simumak. Este simula la cabina de un automóvil con todos los componentes del mismo, además simula escenarios, calles, otros vehículos, condiciones climatológicas etc. Actualmente no se tiene información acerca de las herramientas y la metodología utilizada para el desarrollo del simulador debido a que es propietario. Se puede visitar el sitio a través de http://simumak.com/es/simescar.

Para el uso de Simescar se debe comprar tanto el software como el hardware, ya que de esa forma lo comercializa el propietario del producto, Simumak. El mismo no nos sirve pues el simulador que se propone se debe adaptar a la tecnología existente en la UCI.

Simulador de procesos:

HYSYS es un programa interactivo enfocado a la ingeniería de procesos y la simulación, que se puede utilizar para solucionar toda clase de problemas relacionados con procesos químicos. Este simulador cuenta con una interfaz muy amigable para el usuario, además de permitir el empleo de operadores lógicos y herramientas que facilitan la simulación de diversos procesos. Fue adquirido por AspenTech en el 2004 por lo que es desarrollado en la actualidad por Aspen Technology. Es un simulador bidireccional, ya que el flujo de información va en dos direcciones (hacia delante y hacia atrás). De esta forma, puede calcular las condiciones de una corriente de entrada a una operación a partir de las correspondientes a la corriente de salida sin necesidad de cálculos iterativos. Posee un entorno de simulación modular tanto para estado estacionario como para régimen dinámico. Es un software para la simulación de plantas petroquímicas y afines (Pérez, 2011).

HYSYS es un simulador de tipo comercial el cual necesita de licencia para su utilización, el mismo no permite reutilización de su código ni modificación del mismo, solo puede ser utilizado en la rama de la química, por lo tanto no sirve como solución de la problemática propuesta, la creación de un simulador de interbloqueos par la asignatura de SO.

Utilizar simuladores en el PEA de la UCI unido a la creación de LV da una ventaja significativa en cuanto al aprendizaje del estudiante en la asignatura de SO. El uso de simuladores en los LV además ahorraría presupuesto a la universidad y dotaría tanto al docente como al estudiante de herramientas sofisticadas y altamente didácticas con el objetivo de aumentar la calidad de las clases. En la investigación realizada no se encontraron simuladores gratuitos que den solución a la problemática propuesta, de aquí la importancia de desarrollar un simulador para uso didáctico en la UCI de forma gratuita.

1.4 Metodología a utilizar en el desarrollo de software

Un proceso de desarrollo de software es un método de organizar las actividades relacionadas con la creación, presentación y mantenimiento de los sistemas de software (Larman, 1999).

Un proceso define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. En la ingeniería de software el objetivo es construir un software o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia, reduce el riesgo y hace el proyecto más predecible (Jacobson, y otros, 2000)

1.4.1 Rational Unified Process

RUP es una metodología dirigida por casos de usos, centrado en la arquitectura, y es iterativo e incremental.

Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema ((RUP), Rational Unified Process, 2009)

En RUP los casos de usos guían al sistema por todo su ciclo de vida, desde la toma de requisitos, el diseño, la implementación hasta las pruebas. A través de los casos de usos se crean los modelos del análisis y el diseño que luego dan lugar a la correcta implementación del sistema.

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo ((RUP), Rational Unified Process, 2009)

La arquitectura de un sistema involucra los aspectos dinámicos y estáticos más significativos del mismo. De aquí que se tome una correcta toma de decisiones sobre cómo debe ser construido el sistema. Además los elementos de la arquitectura influyen directamente en la calidad del sistema,

rendimiento, reutilización, seguridad, etc.

Según ((RUP), Rational Unified Process, 2009) el equilibrio correcto entre los casos de uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

RUP posee cuatro fases y varios flujos de trabajo lo cual organiza y controla la construcción del sistema desde su edad más temprana hasta su despliegue. Además RUP cuenta con vasta documentación y guías para su aprendizaje. Otras ventajas de utilizar RUP ((RUP), Rational Unified Process, 2009):

- Gestión de requisitos.
- Desarrollo de software iterativo.
- Desarrollo basado en componentes.
- Modelado visual (usando UML).
- Verificación continua de la calidad.
- Gestión de cambios.

Se decide utilizar RUP como metodología de desarrollo por ser iterativo e incremental, centrado en la arquitectura y guiado por casos de usos, además es eficiente para proyectos extensos y complejos. El cumplimiento de hitos al final de cada fase así como la asignación de roles para el cumplimiento de esos hitos garantiza el desarrollo del software de manera efectiva y eficiente, asegurando toda la documentación necesaria para la elaboración del sistema.

A continuación se especifican los artefactos a generar durante los flujos de trabajo Modelamiento de negocio, Requisitos y Análisis y Diseño propuestos por RUP.

En el Modelamiento de Negocio se decide generar:

- Modelo de dominio.

- Descripción del modelo de dominio.

En el flujo de trabajo de requisitos se elaborará:

- Especificación de requisitos de software.
- Matriz de trazabilidad.
- Prototipos de interfaz de usuario.
- Lista de chequeo de especificación de requisitos.
- Métricas de la calidad de la especificación.
- Diagrama de casos de uso del sistema.
- Descripción de casos de uso del sistema.
- Métricas para la validación de los casos de uso de sistema.

En Análisis y Diseño:

- Diagrama de clases de análisis.
- Vista lógica de arquitectura.
- Diagrama de casos de uso arquitectónicamente significativos.
- Diagrama de clases de diseño.
- Diagramas de secuencia del diseño.
- Métricas de tamaño de clase.

Con la elaboración de los artefactos mencionados en los diferentes flujos de trabajo se asegurará la construcción del simulador en flujos de trabajos posteriores.

1.5 Lenguaje de modelado

Un lenguaje nos permite comunicarnos sobre un tema. En el desarrollo del sistema, el tema incluye los requisitos y el sistema. Sin un lenguaje, es difícil para los miembros del equipo de comunicación y colaboración desarrollar con éxito un sistema (Alhir, 2003).

1.5.1 Lenguaje de modelado unificado (UML) 2.1

UML es un lenguaje visual para el modelado y la comunicación acerca de los sistemas mediante el uso de diagramas y texto de apoyo (Alhir, 2003).

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y unas reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema. Este lenguaje nos indica cómo crear y leer los modelos, pero no dice cómo crearlos. Esto último es el objetivo de las metodologías de desarrollo. Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones (Orallo, 2002).

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes su construcción.
- Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden ser utilizados para su futura revisión.

Se decide utilizar UML 2.1 como lenguaje de modelado pues está estrechamente relacionado con la metodología de desarrollo a utilizar RUP. UML es un lenguaje bastante popular entre las industrias de desarrollo de software a nivel mundial y en la UCI por las distintas ventajas que propone su uso. Además de tener una vasta documentación y guías para su empleo, cuenta con diversos cursos que facilitan su enseñanza y posterior empleo.

1.6 Herramienta de modelado (CASE)

CASE (Cumputer Aided Software Engineering).Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de software (Morales, 2011).

Según (Sommerville, 2005) comprende un amplio abanico de diferentes tipos de programas que se utilizan para ayudar a las actividades del proceso del software, como análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas. En la actualidad, todos los métodos vienen con tecnología CASE asociada, como los editores para las notaciones utilizadas en el método y módulos de análisis que verifican el modelo del sistema según las reglas del método y generadores de informes que ayudan a crear la documentación del sistema. Las herramientas CASE también incluyen un generador de código que automáticamente genera código fuente a partir del modelo del sistema y de algunas guías de procesos para los ingenieros de software.

1.6.1 Visual Paradigm para UML 8.0

Visual Paradigm es una de las herramientas UML CASE del mercado, considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelente facilidades de interoperabilidad con otras aplicaciones.

Fue creada para el ciclo vital completo del desarrollo de software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de las clases.

Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros (Morales, 2011).

Características:

Visual Paradigm es un producto de alta calidad que soporta aplicaciones web, varios idiomas, genera código para Java y exporta como HTML. Es un producto fácil de instalar y actualizar, mantiene la compatibilidad entre sus ediciones. Además se integra con las siguientes herramientas Java (Eclipse, Jbuilder, NetBeans IDE, Oracle Jdeveloper, BEA Weblogic, etc.).

Algunas de las ventajas de utilizar Visual Paradigm son las siguientes (Morales, 2011):

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requisitos y de especificación de componentes.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario .Net.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Desventajas:

- Las imágenes y reportes generados, no son de muy buena calidad (esta desventaja se reduce total o parcialmente en la versión 8.0 de Visual Paradigm).
- Es software propietario.

Se decide utilizar Visual Paradigm 8.0 debido a todas las ventajas mencionadas anteriormente, además podemos agregar que está perfectamente integrado con UML como lenguaje de modelado además de soportar RUP como metodología de desarrollo, escogidas anteriormente para la modelación del sistema. Por otro lado y como una de las ventajas más importantes es que la UCI posee la licencia para poder trabajar con Visual Paradigm.

1.7 Ingeniería de Requisitos (IR)

La ingeniería de requisitos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, que es lo que el cliente quiere y como interactuarán los usuarios finales con el software.

La IR, como todas las demás actividades de la ingeniería de software, debe adaptarse a las necesidades del proceso, el proyecto, el producto y las personas que realizan el trabajo. Desde la perspectiva del proceso de software, la IR es una acción de la Ingeniería de Software que comienza durante la actividad de comunicación y continúa en la actividad de modelado (Pressman, 2005).

¿Ahora, que se entiende por requisitos en la Ingeniería de Software?

Según (Sommerville, 2005) los requisitos de un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar una información. El proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se denomina Ingeniería de requisitos.

Separa los requisitos en dos grandes grupos:

Los requisitos del usuario son declaraciones, en lenguaje natural y en diagramas, de los servicios que se espera que el sistema proporcione y de las restricciones bajo las cuales debe funcionar.

Los requisitos del sistema establecen con detalle las funcionalidades, servicios y restricciones operativas del sistema. Los requisitos de sistemas de software son clasificados en requisitos funcionales y no funcionales o requisitos de dominio (Sommerville, 2005).

Requisitos funcionales:

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe relacionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden aclarar explícitamente lo que el sistema no debe hacer.

Requisitos no funcionales:

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema.

Requerimientos del dominio:

Son requerimientos que provienen del dominio de aplicación del sistema y que reflejan las características y restricciones de ese dominio. Pueden ser funcionales o no funcionales.

La ingeniería de requisitos es de suma importancia a la hora de construir un producto de calidad a la altura de las necesidades del cliente. Los requisitos son la base de la construcción del sistema y deben ser lo más cercano a lo que el cliente necesita. Por eso la ingeniería de requisitos está entre los procesos más importantes en la construcción de un sistema.

1.7.1 Actividades de la Ingeniería de Requisitos

Existen cuatro actividades básicas (extracción, análisis, especificación y validación) que se tienen que llevar a cabo para completar el proceso. Estas actividades ayudan a reconocer la importancia que tiene, para el desarrollo de un proyecto de software, realizar una especificación y administración adecuada de los requisitos de los clientes o usuarios (Hung, 2011).

Extracción: esta fase representa el comienzo de cada ciclo. Extracción es el nombre comúnmente dado a las actividades involucradas en el descubrimiento de los requisitos del sistema.

Análisis: sobre la base de la extracción realizada previamente, comienza esta fase. Usualmente se hace un análisis luego de haber producido un bosquejo inicial del documento de requisitos; aquí se leen los requisitos, se conceptúan, se investigan, se intercambian ideas con el resto del equipo, se resaltan los problemas, se buscan alternativas y soluciones, y luego se van fijando reuniones con el cliente para discutir los requisitos.

Especificación: en esta fase se documentan los requisitos acordados con el cliente, en un nivel apropiado de detalle. En la práctica, esta etapa se va realizando conjuntamente con el análisis, pero se podría decir que la Especificación es el “pasar en limpio” el análisis realizado previamente aplicando técnicas y/o estándares de documentación, como la notación UML.

Validación: la validación es la etapa final de la IR. Su objetivo es verificar todos los requisitos que aparecen en el documento especificado para asegurarse que representan una descripción, por lo menos, aceptable del sistema que se debe implementar. Esto implica verificar que los requisitos sean consistentes y que estén completos.

En algunas bibliografías se puede encontrar además la gestión de requisitos que no es más que el seguimiento que se le hace a los requisitos a través de las etapas anteriores, y el objetivo es realizar algunas actividades que permitan identificar, seguir y controlar los cambios sufridos en cualquier momento del ciclo de vida. En el caso de la toma de requisitos para el desarrollo del sistema se realizó la gestión de requisitos a lo largo de la ingeniería de requisitos.

1.7.2 Herramientas de la Ingeniería de Requisitos

Existen diversas técnicas y herramientas que se utilizan para llevar a cabo cada una de las actividades del proceso de Ingeniería de Requisitos, una de las razones por las cuales surgen los errores a la hora del levantamiento es la existencia de una gama de herramientas. No existe una especie de guía para el uso de los desarrolladores, estos utilizan incluso en la captura más de una técnica en cada de las actividades que contiene el proceso (Hung, 2011).

Herramientas utilizadas para la extracción de los requisitos.

Entrevistas y cuestionarios: las entrevistas y cuestionarios se emplean para reunir información proveniente de personas o grupos, información que se obtiene conversando con el encuestado. Las preguntas suelen distinguirse en dos categorías: abiertas y cerradas. Las preguntas abiertas permiten que los encuestados respondan con su propia terminología, mientras que las preguntas cerradas predeterminan todas las posibles respuestas y el interrogado elige entre las opciones presentadas.

Brainstorming (tormenta de ideas): este es un modelo que se usa para generar ideas. La intención en su aplicación es la de generar la máxima cantidad posible de requisitos para el sistema. No hay que detenerse en pensar si la idea es o no del todo utilizable.

Casos de uso: es una técnica muy usada en la extracción, donde se agrupan los requisitos por funcionalidades para luego describir las acciones del usuario y la respuesta del sistema, o sea, la secuencia de acciones que sigue para lograr un objetivo. Esta técnica sirve para todas las etapas de la ingeniería de requisitos desde la extracción hasta la validación incluyendo la gestión de requisitos.

Las herramientas utilizadas para la validación de los requisitos:

Validación por prototipos ya que es una técnica utilizada para mostrarle al cliente o usuario una propuesta de interfaz de las funcionalidades. Se emplea para lograr una mejor comprensión de los requisitos y consiste en diseñar una propuesta de interfaz de un requisito determinado que debe ir incluido en el producto final, aunque esta puede presentar modificaciones.

La matriz de trazabilidad de los requisitos es una de las técnicas más utilizadas tanto para validar como para gestionar los requisitos, la misma muestra las relaciones entre los casos de usos y los requisitos del usuario, aspecto importante a tener en cuenta en la ingeniería de requisitos.

1.8 Modelos del sistema

Los modelos del sistema son representaciones gráficas que describen los procesos del negocio, el problema a resolver y el sistema que tiene que ser desarrollado. Debido a las representaciones gráficas usadas, los modelos son a menudo más comprensibles que las descripciones detalladas en lenguaje natural de los requisitos del sistema. Ellos constituyen también un puente importante entre el proceso de análisis y diseño (Sommerville, 2005).

1.8.1 Patrones de casos de uso

Un patrón es un problema/solución que estandariza principios y sugerencias relacionadas frecuentemente con la asignación de responsabilidades. Es la representación de reiterados problemas en un tema determinado (Larman, 1999).

En la realización del diagrama de casos de uso del sistema se utilizan un conjunto de patrones que facilitan su representación. Estos patrones son:

Inclusión (Inclusion): inclusión es un patrón de estructura. Consiste en dos casos de uso y una relación de inclusión entre el caso de uso base y el caso de uso incluido. El caso de uso base puede ser concreto o abstracto. Se utiliza este patrón cuando un flujo de datos puede ser incluido en el flujo de otro caso de uso.

Extensión (Extension): extensión es un patrón de estructura. Consiste en dos casos de uso y una relación extendida entre ellos. Un caso de uso es extendido en el caso de uso base. El referente puede ser concreto o abstracto. Este patrón se aplica cuando un flujo extiende el flujo de otro caso de uso.

Concordancia (Commonality) Especialización: es un patrón de concordancia que contiene casos de uso del mismo tipo. En este caso, estos son modelados como una especialización de casos de uso de tipo de uso común. Todas las acciones en estos casos de uso son heredadas por los casos de uso hijos, donde otras acciones serán adicionadas o acciones heredadas que serán especializadas. Este patrón es aplicable cuando la utilización de los casos de uso que han sido modelados son del mismo tipo, y este tipo debe hacerse visible en el modelo.

1.9 Conclusiones

Con la realización del capítulo se pudo obtener una visión más clara acerca del tema de interbloqueos y los principales factores que interactúan en el mismo. Igualmente se realizó un estudio acerca de los laboratorios virtuales y la simulación en la educación para de esta forma tener una idea más precisa de lo que se debía hacer y cómo se debía hacer. Además se hizo una selección de herramientas, tecnologías y técnicas entre las que se incluyeron RUP como metodología de desarrollo, UML en su versión 2.1 como lenguaje de modelado y Visual Paradigm en su versión 8.0 como herramienta de modelado, de manera que permitieran un desarrollo eficiente y eficaz de la solución propuesta.

Capítulo II: Descripción de la solución propuesta

En este capítulo se describe el funcionamiento general del negocio mediante un modelo de dominio. Además, se identifican los requisitos funcionales y los nos funcionales de la solución propuesta. Se describen los actores y los casos de uso del sistema, así como sus relaciones, reflejados perfectamente en el modelo de casos de usos. Además se realiza el análisis del sistema según la metodología de desarrollo escogida en el cual se realizan los diagramas de clases de análisis de los casos de uso. Se selecciona la arquitectura candidata con los estilos arquitectónicos correspondientes, se definen las vistas lógicas de la arquitectura del laboratorio virtual y del simulador de interbloqueo, así como los casos de uso arquitectónicamente significativos. Para dar solución al diseño se realiza el diagrama de casos de uso del diseño, se describen los patrones de diseño utilizados y se elabora el diagrama de secuencia correspondiente.

2.1 Modelo de Dominio

La metodología de desarrollo RUP propone dos opciones para realizar el modelado del negocio de un sistema, por casos de uso del negocio y por clases de dominio o modelo de dominio.

Según (Jacobson, y otros, 2000) un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio.

El modelo del dominio se describe mediante diagramas de UML (especialmente mediante diagramas de clases).

Estos diagramas muestran a los clientes, usuarios, revisores y a otros desarrolladores las clases del dominio y cómo se relacionan unas con otras mediante asociaciones.

¿Por qué usar modelo del dominio?

Como no se tiene una perspectiva clara sobre el funcionamiento del proceso de negocio o no se tienen los elementos suficientes (procesos y roles del proceso de negocio) para elaborar un modelo de negocio entonces es decidió utilizar modelo de dominio.

2.1.1 Conceptos del Modelo de Dominio

- Laboratorio Virtual: es un sistema computacional que pretende aproximar el ambiente de un laboratorio tradicional.
- Subsistema de ejercicios: subsistema del Laboratorio Virtual de Sistemas Operativos encargado de gestionar todo lo referente a las formas posibles en que el estudiante puede evaluar sus conocimientos.
- Simulador: es un programa informático que imita el funcionamiento de un sistema real durante un intervalo de tiempo.
- Algoritmo: tipo de modelo científico, que emplea fórmulas matemáticas para estudiar y resolver comportamientos de un sistema complejo o difícil de observar en la realidad.
- Procesos: un proceso es un programa en ejecución con un estado o contexto de ejecución.
- Recursos: un recurso puede ser cualquier elemento (archivo, segmento de memoria, procesador, variable, etc.) que solo pudiera ser utilizado por un proceso a la vez.
- Grafo: representación visual de un conjunto de nodos relacionados entre sí por vectores.
- Nodos: representación visual de procesos o recursos con sus instancias.
- Vectores: representación visual de la relación que existe entre los nodos.

2.1.2 Modelo de Dominio

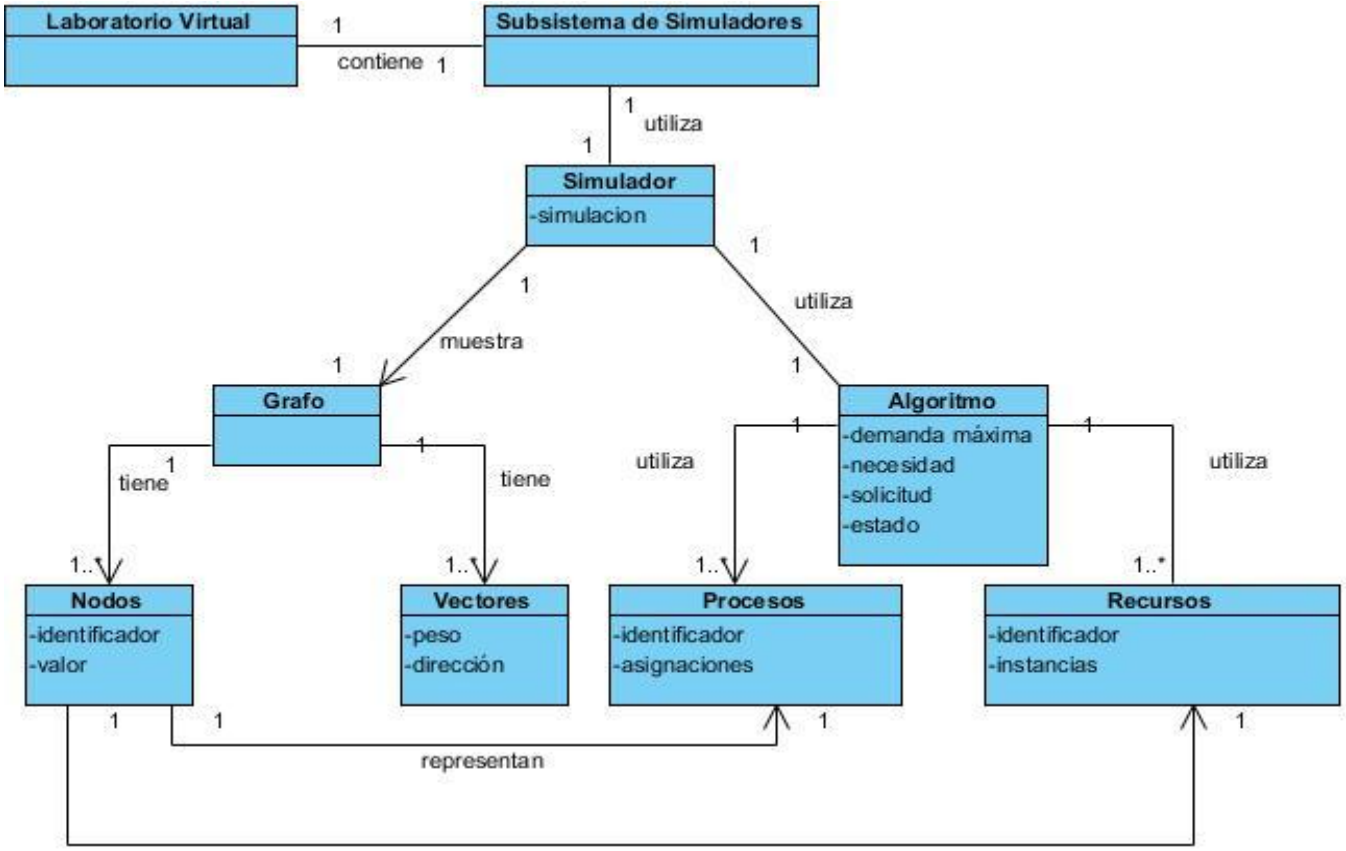


Figura 1 Modelo del Dominio.

2.1.2 Descripción del Modelo de Dominio

El Laboratorio Virtual de SO cuenta con un subsistema de ejercicios, el cual a su vez tiene un simulador. El simulador utiliza un algoritmo que determina si algún proceso puede utilizar algún recurso de manera que no genere un estado de interbloqueo y de generarse, como recuperarse del mismo. Para una mejor comprensión del problema la simulación es graficada donde un número finito de procesos compiten por instancias de un grupo de recursos dados (nodos ambos) relacionados por un vector dirigido: si la dirección es del nodo proceso al nodo recurso, el proceso está solicitando una o varias instancias del recurso en cuestión, si la dirección es del nodo recurso al nodo proceso, el recurso está asignando una o varias instancias al proceso.

2.2 Requisitos del sistema

Un requisito es una característica de diseño, una propiedad, condición, capacidad o un comportamiento esperado de un sistema. Los requisitos constituyen la descripción de los deseos o de las necesidades de un cliente (Jacobson, y otros, 2000).

2.2.1 Requisitos funcionales

Son requisitos que especifican las acciones que debe ser capaz de realizar el sistema, sin considerar restricciones físicas; requisitos que especifican comportamiento de entrada/salida de un sistema (Larman, 1999).

Requisitos Funcionales	
RF1	Inicializar simulación.
RF2	Introducir datos.
RF3	Generar inicialización.
RF4	Cargar simulación.
RF5	Realizar simulación.
RF6	Mostrar consola.
RF7	Mostrar matrices y vectores.
RF8	Navegar por simulación.
RF9	Mostrar estado de simulación.
RF10	Visualizar grafo animado.
RF11	Modificar proceso.

RF12	Modificar recurso.
RF13	Guardar simulación.
RF14	Agregar vector solicitud.

Tabla 1 Requisitos funcionales.

2.2.2 Requisitos no funcionales

Según (Larman, 1999) los requisitos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, fiabilidad etc. Es un requisito que especifica restricciones físicas sobre un requisito funcional.

Apariencia o interfaz externa:

- RNF1: el diseño de la interfaz debe ser sencillo y fácil de usar, la combinación de colores estará en correspondencia con los colores del laboratorio virtual de manera que mantenga la uniformidad con el mismo. Debe contar con los logos que identifique a la UCI así como al laboratorio virtual.

- RNF2: debe contar con iconos que ilustren de forma lógica las funciones que representan. Además debe permitir al usuario realizar acciones de una manera rápida, minimizando los pasos a dar en cada proceso, reduciendo de esta forma la complejidad en su uso.

- RNF3: el sistema tendrá las funcionalidades más importantes en la interfaz principal. El usuario no tendrá que navegar por varias secciones para ver los elementos de interés para él. Cada sección tendrá los elementos y descripciones necesarias para orientar al usuario.

- RNF4: la información del simulador será presentada de forma clara y organizada, permitiendo una correcta interpretación por parte de los usuarios.

Usabilidad:

- RNF5: el sistema contará con una interfaz que permita la fácil navegación tanto para usuarios expertos (profesores y estudiantes de la asignatura) como para los que no tienen conocimientos profundos del tema (otras personas ajenas a la asignatura).

Rendimiento:

- RNF6: la respuesta a solicitudes más complejas de los usuarios del sistema no debe exceder de 9 segundos.

- RNF7: la aplicación deberá estar disponible las 24 horas del día.

Portabilidad:

- RNF8: debe poder ejecutarse tanto en Sistemas Operativos desde Windows XP Profesional Service Pack 1 o superior como en Sistemas Operativos GNU/Linux.

Hardware:

- RNF9: tarjeta de memoria RAM de 256 MB o superior.

- RNF10: procesador Pentium IV a 3000 MHz o superior

- RNF11: computadora cliente con capacidad de 50 Mb de disco duro como mínimo.

Ayuda del sistema

- RNF12: el sistema contará con una ayuda que constituirá una guía de apoyo en el momento de hacer uso de la aplicación. Esta describe todas las funcionalidades del sistema.

2.3 Modelo de Casos de uso del sistema

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan a los requisitos funcionales. Todos los casos de usos juntos constituyen el modelo de casos de uso el cual describe la funcionalidad total del sistema (Larman, 1999).

2.3.1 Actores del sistema

Un actor es una persona, grupo de personas, entidad u otro sistema que interactúe con el sistema. Una vez definidos los actores, se puede identificar el entorno externo del sistema.

Actor	**Descripción**
Usuario	Un usuario es aquel que realiza cualquier operación dentro del sistema. Puede ser un estudiante o un profesor que requiera del uso del simulador y cuente con el conocimiento necesario en el tema.

Tabla 2 Actores del sistema.

2.3.2 Diagrama de casos de uso del sistema

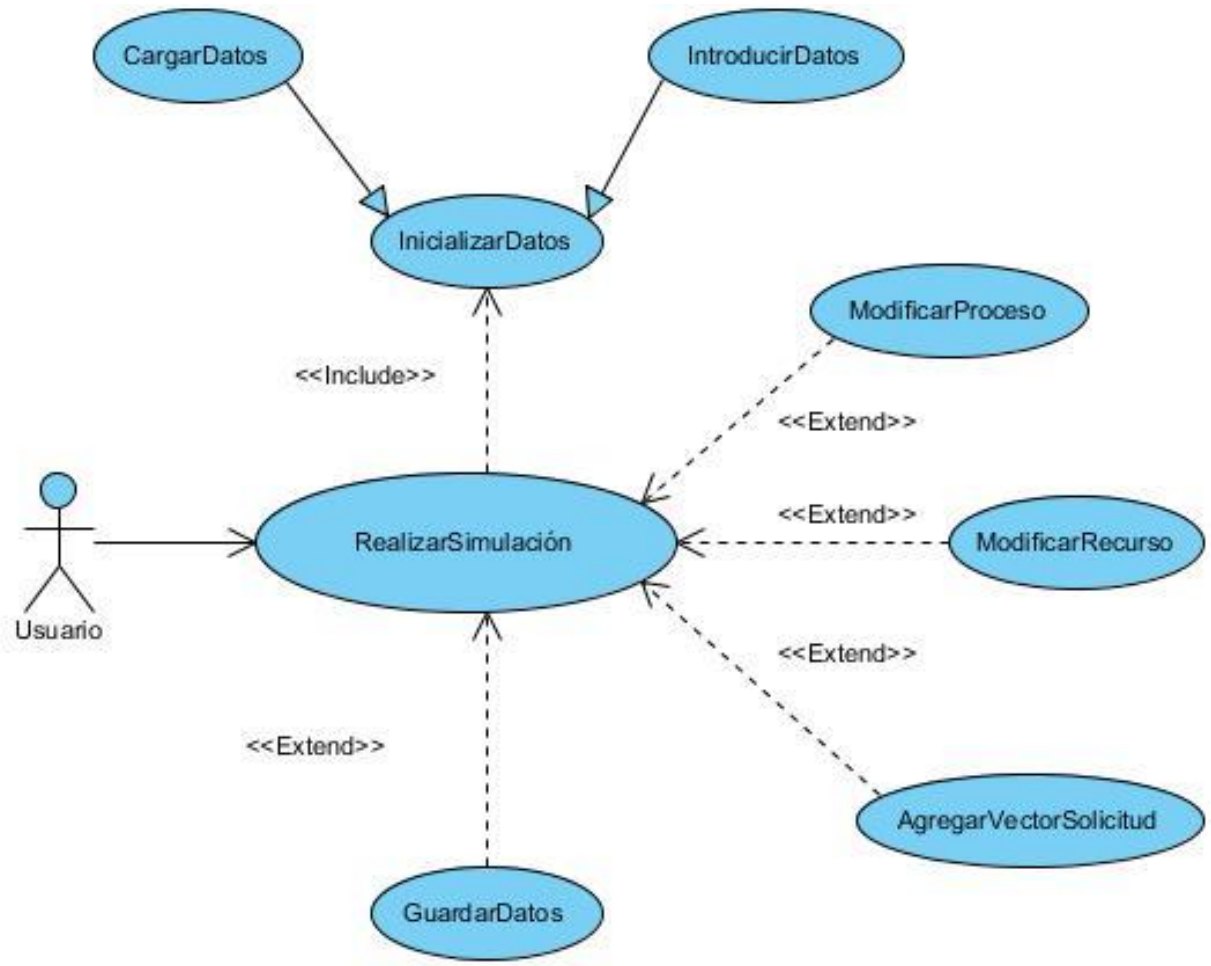


Figura 2 Diagrama de casos de uso del sistema.

Para el usuario realizar una simulación primero debe inicializar los datos de la misma. Para iniciar los datos tiene dos variantes, introducir los datos (manuales o generarlos automáticamente) o cargarlos de un fichero previamente guardados. Una vez cargado los datos de una manera u otra, inicializa la simulación, ya en la simulación el usuario puede modificar las instancias de los recursos y las peticiones de los procesos. Además el usuario puede navegar en la simulación y ver las variaciones en las variables de la simulación. Como se dijo anteriormente el usuario puede guardar una simulación en curso o finalizada.

2.3.3 Descripción de los casos de uso del sistema

La descripción de los casos de uso del sistema describe paso a paso la forma en que interactúa el sistema y el usuario.

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema (Figueredo Jiménez, y otros, 2010).

Descripción del caso de uso Introducir Datos.

<table>
<tr><td>Caso de uso:</td><td colspan="2">Introducir Datos</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario introduce los datos de la simulación tanto de forma manual como de forma automática.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2"></td></tr>
<tr><td>Referencias</td><td colspan="2">RF1, RF2, RF3</td></tr>
<tr><td>Prioridad</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo normal de eventos</td></tr>
<tr><td colspan="3">Sección “Introducción de los datos de forma manual”</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1. El usuario selecciona la opción Nueva simulación en el menú “Archivo”.</td><td colspan="2">2. El sistema muestra los formularios para inicializar los datos y activa el formulario de los recursos.

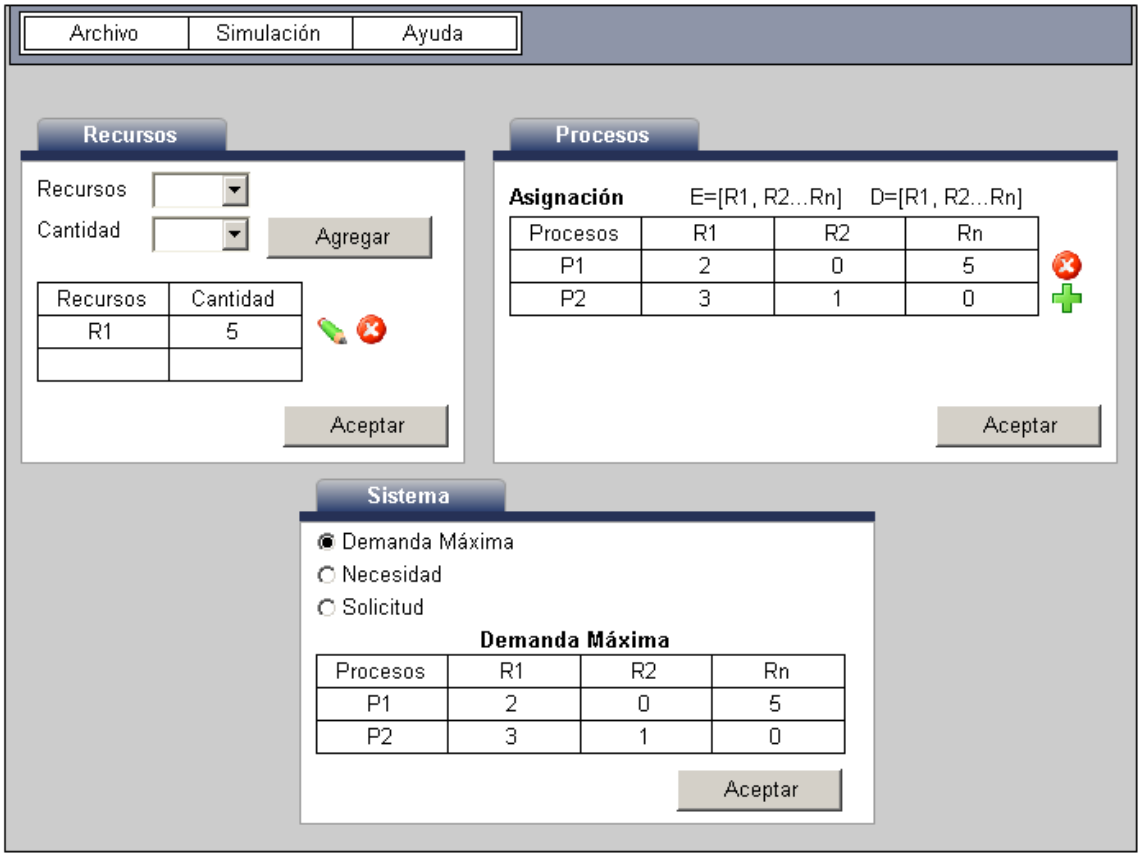
El formulario muestra:

- Un campo para seleccionar el</td></tr>
</table>

	recurso. - Un campo para entrar la cantidad de instancias del recurso. - Una lista de recursos con sus instancias y las opciones de modificar y eliminar un recurso. - Un botón para agregar un recurso a la lista. - Un botón para agregar los recursos al sistema.
3. El usuario entra un recurso así como sus instancias, y presiona el botón “Agregar” recurso.	4. El sistema agrega el recurso a la lista de recursos. Si el usuario desea seguir agregando recursos ir al paso 3 del flujo normal de eventos.
5. El usuario presiona el botón “Aceptar” Si el usuario desea eliminar un recurso debe presionar el botón eliminar ilustrado con una “X” al lado del recurso correspondiente. Si el usuario desea modificar un recurso debe presionar el botón editar ilustrado con una “lápiz” al lado del recurso correspondiente.	6. Los recursos son agregados al sistema. El sistema activa el formulario para introducir los procesos y las asignaciones de cada proceso. El formulario muestra: - Una lista con los recursos y sus instancias. - Una tabla con los procesos y los campos para introducir los recursos. - Un botón para agregar un nuevo proceso a la tabla. - Un botón para eliminar un proceso

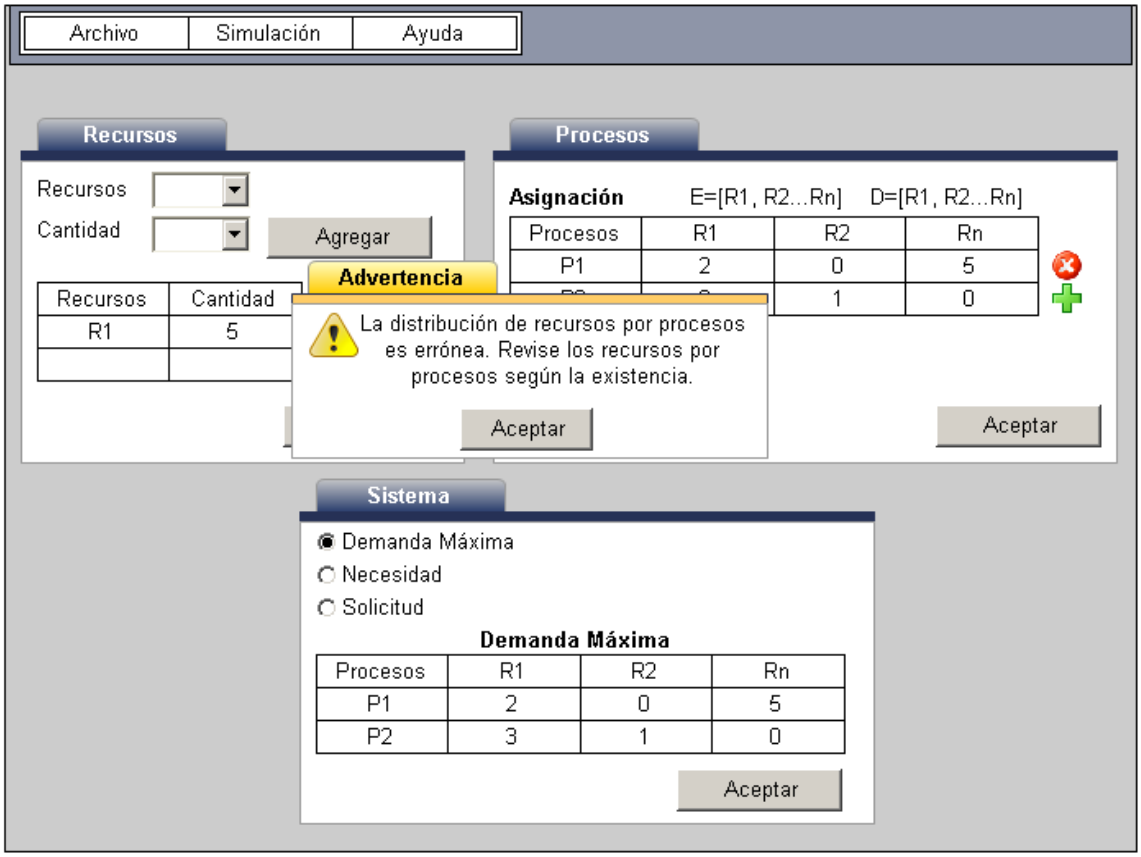
	de la tabla. - Un botón para agregar los procesos al sistema.
7. El usuario entra los procesos y sus asignaciones al sistema y presiona el botón Agregar proceso ilustrado con un “+”. Si el usuario desea modificar un proceso debe tener en cuenta que ningún proceso esté utilizando instancias de ese recurso y debe presionar el botón ir a recursos ver evento 2 del flujo normal de eventos.	8. El sistema agrega un nuevo proceso en blanco a la tabla. Si el usuario desea entrar otro proceso ir al paso 7 del flujo normal de eventos.
9. El usuario presiona el botón “Aceptar” Si el usuario desea eliminar un proceso debe presionar el botón eliminar ilustrado con una “X” al lado del proceso correspondiente.	10. Los procesos son agregados al sistema. El sistema activa el formulario para introducir las matrices del sistema. El formulario muestra: - Un grupo de opciones (Demanda máxima, necesidad y solicitud) para que el usuario seleccione el criterio por el cual se hará la simulación - Una tabla de entrada de datos según el criterio seleccionado. - Un botón para comenzar con la simulación.

11. El usuario selecciona del conjunto el criterio de simulación que desea utilizar. Llena la matriz con los valores a su conveniencia y presiona el botón “Aceptar”.	12. La matriz es agregada al sistema según el criterio seleccionado.
	13. Ver Descripción del caso de uso Inicializar Datos



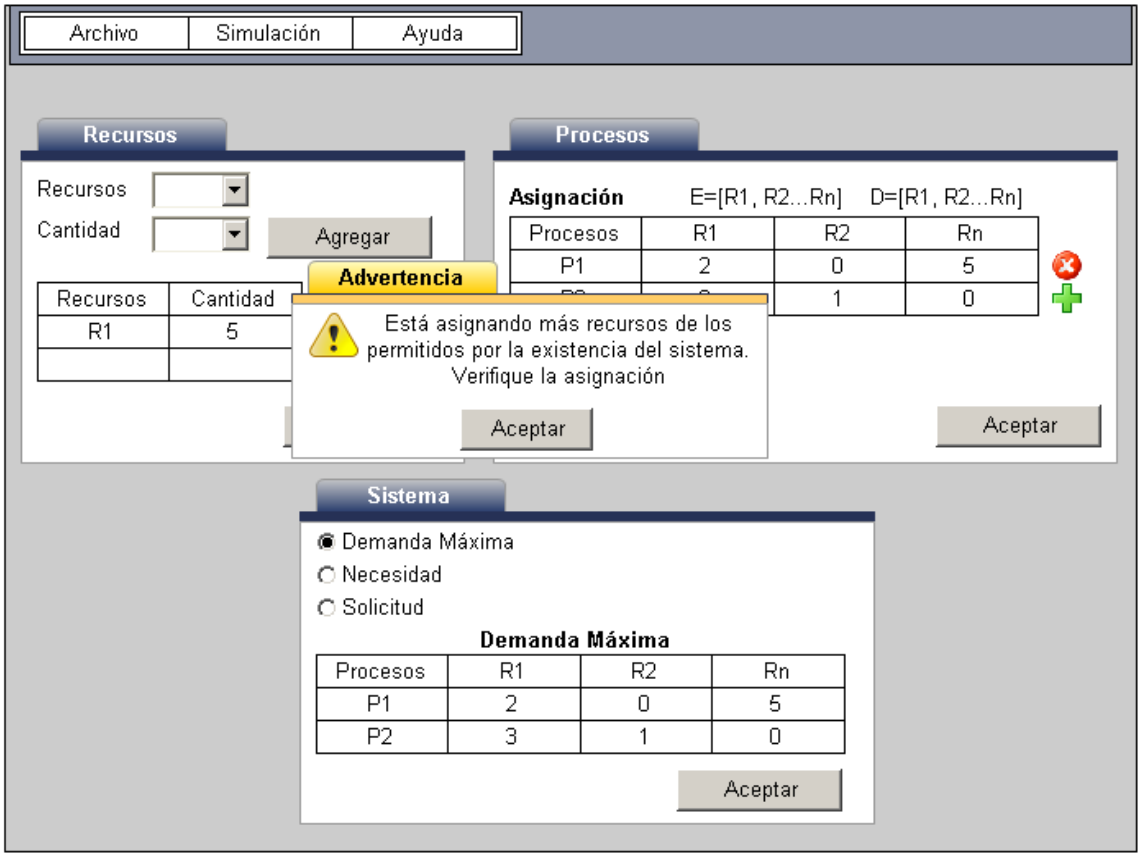
Flujo alterno insertar procesos	
Acción del actor	**Respuesta del sistema**
	9.1. El sistema muestra un mensaje de notificación al usuario informándole que la distribución de recursos por procesos es

	errónea.



Flujo alterno Insertar Sistema

Acción del actor	**Respuesta del sistema**
	11.1. El sistema muestra un mensaje de notificación al usuario informándole que está asignando más recursos de los permitidos por la existencia del sistema.



Poscondiciones	

***Sección* "Introducción de los datos de forma automática"**	
Acción del actor	**Respuesta del sistema**
El usuario selecciona la opción generar datos automáticos	El sistema genera los recursos así como sus instancias de forma aleatoria
	El sistema genera los procesos así como asignaciones y las solicitudes de forma aleatoria.
	El sistema genera el criterio para realizar la simulación de forma aleatoria.

	Ver Descripción del caso de uso Inicializar Datos

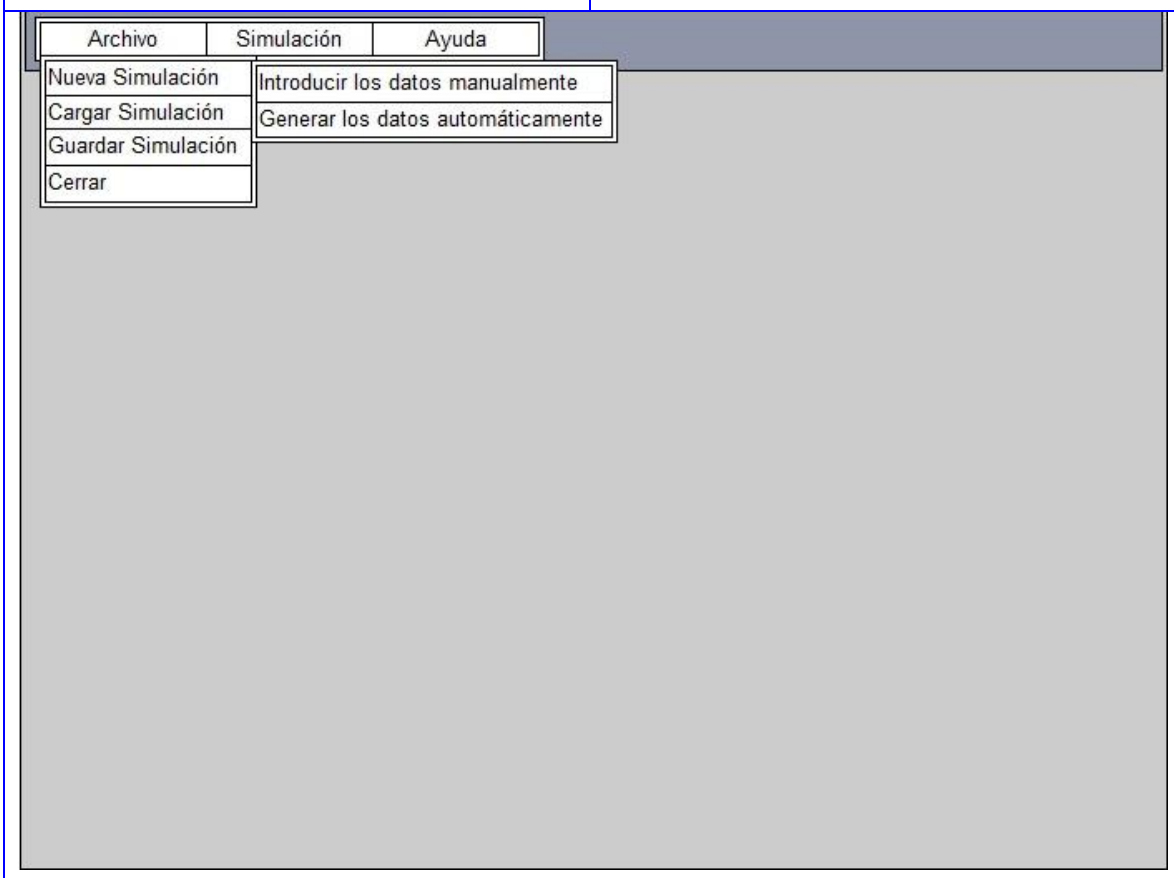


Tabla 3 Descripción de casos de uso Introducir Datos.

Descripción del caso de uso Inicializar Datos.

Caso de Uso:	Inicializar Datos
Actores:	Usuario
Resumen:	Se inicializan los datos cargados tanto de una nueva simulación, como de una carga de fichero de una simulación guardada.
Precondiciones:	
Referencias	RF1

<table>
<tr><td>Prioridad</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Inicializar Datos”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td></td><td colspan="2">1. El sistema carga el juego de datos entrados por el usuario:
Inicializa la lista de recursos y sus instancias
Inicializa la lista de procesos así como sus asignaciones.
Inicializa la simulación según el criterio seleccionado.
Establece la iteración actual así como los demás datos de la simulación.
Crea los ficheros temporales para almacenar los datos de la simulación.</td></tr>
<tr><td></td><td colspan="2">2. EL sistema va a la vista de realizar simulación luego de haber creado todos los datos para realizar la simulación</td></tr>
<tr><td>Poscondiciones</td><td colspan="2">Deben quedar conformados todos los datos necesarios para comenzar la simulación.</td></tr>
</table>

Tabla 4 Descripción del caso de uso Inicializar Datos.

2.4 Análisis del sistema

El análisis de requisitos genera la especificación de características operacionales de software: indica la interfaz del software con otros elementos del sistema, y establece las restricciones que debe tener el software (Pressman, 2005). Además permite que el ingeniero de software construya los modelos que representan escenarios del usuario, actividades funcionales, clases del problema y sus relaciones a partir de los requisitos básicos definidos durante tareas anteriores.

Durante el análisis, analizamos los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero – incluyendo su arquitectura (Jacobson, y otros, 2000).

2.4.1 Diagrama de clases del análisis

Una clase de análisis representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema. Se centra en el tratamiento de los requisitos funcionales, esto hace que el análisis sea más evidente en el contexto del dominio del problema, más “conceptual”, a menudo de mayor granularidad que sus contrapartidas de diseño e implementación. Las clases del análisis son representadas con tres estereotipos básicos: interfaz, control y entidad (Pressman, 2005).

A continuación se muestran los diagramas de clases del análisis correspondiente a los casos de uso Introducir Datos y Realizar Simulación.

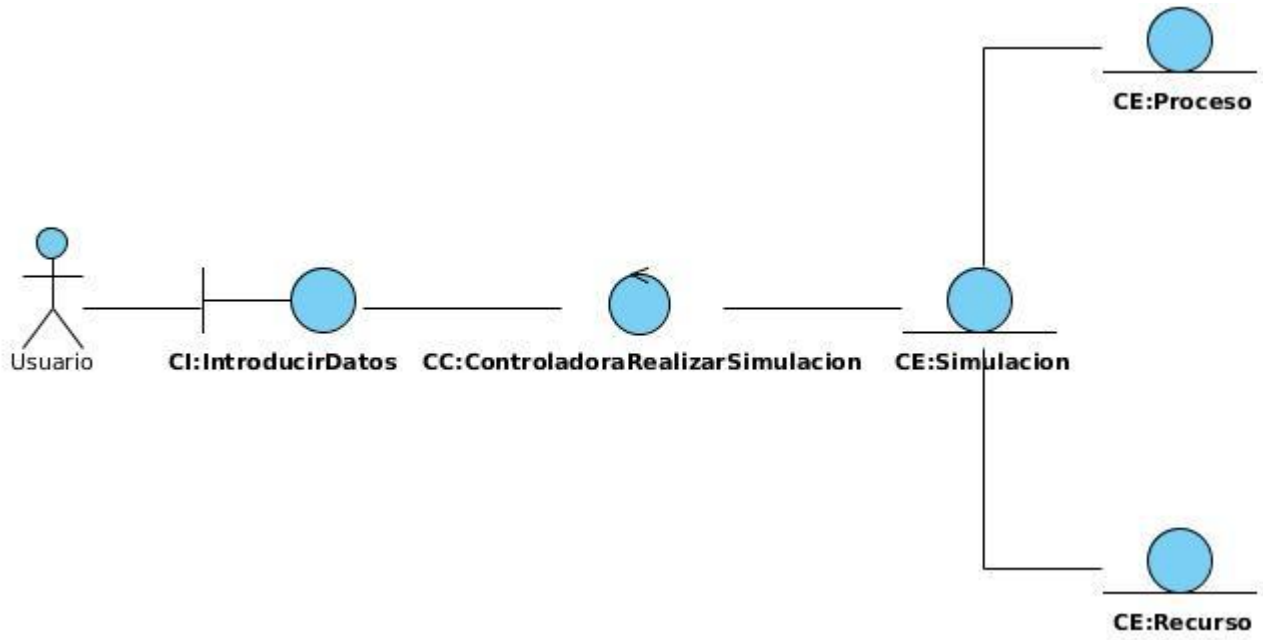


Figura 3 Diagrama de clases del análisis del CU Introducir Datos.

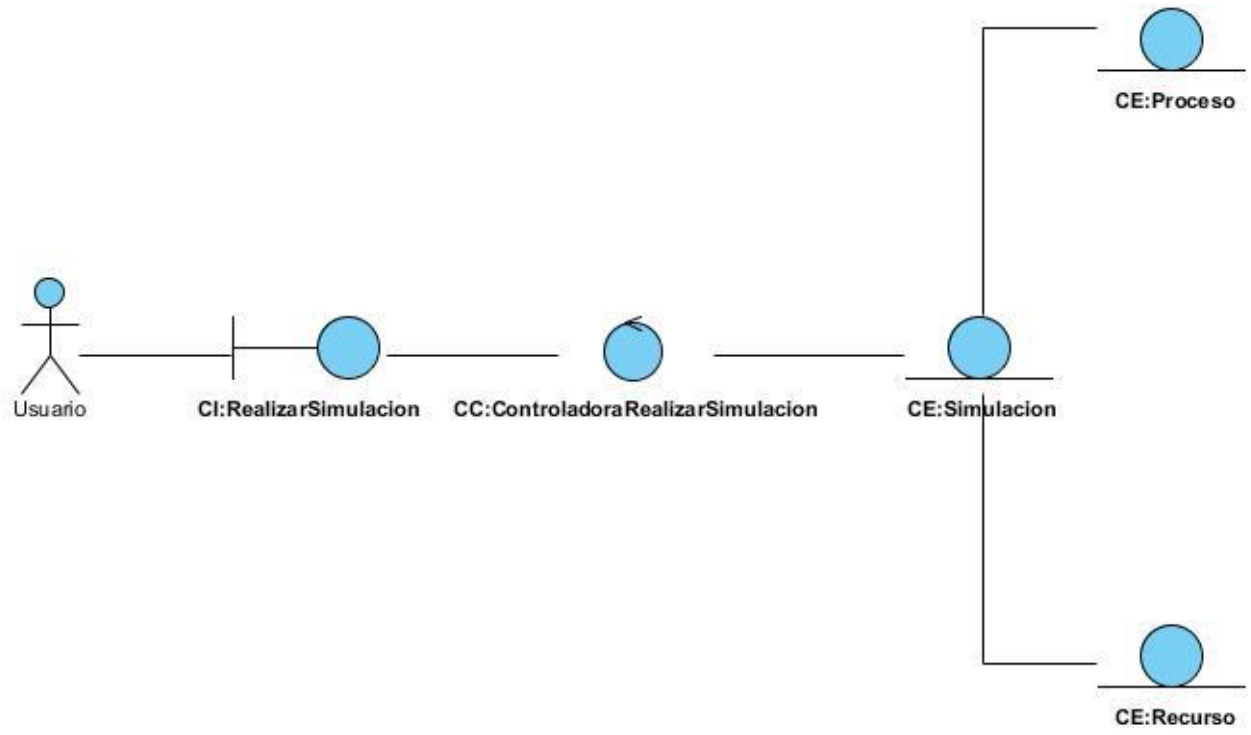


Figura 4 Diagrama de clases de análisis del CU Realizar Simulación.

2.5 Arquitectura de software (AS)

Una arquitectura es el sistema de decisiones significativas sobre la organización de un sistema de software, la selección de los elementos estructurales y de sus interfaces por los cuales el sistema es compuesto, junto con su comportamiento según lo especificado en las colaboraciones entre estos elementos, la composición de estos elementos estructurales y del comportamiento en subsistemas progresivamente más grandes y el estilo arquitectónico que dirige esta organización, los elementos y sus interfaces, sus colaboraciones y su composición (Pressman, 2005).

En el año 2000 la IEEE hace la definición oficial de AS en su documento IEEE 1471, que dice así:

“La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución”.

2.5.1 Estilos Arquitectónicos

Según (C. Suarez, y otros, 2009) los estilos arquitectónicos son uno de los aspectos fundamentales de la disciplina arquitectura de software. Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica donde se conjugan componentes, conectores, configuraciones y restricciones. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales.

Los estilos arquitectónicos son arquitecturas comunes o prototipos de arquitecturas que por sus características pueden ser utilizadas en la construcción de diferentes programas.

Los estilos de arquitectura se definen como las 4 C (E. Perry, y otros, 1992):

- Componentes (Elementos)
- Conectores
- Configuraciones

- Restricciones (Constraints)

Para la selección de los estilos arquitectónicos a utilizar se realizó un estudio de algunos estilos descritos por (Reynoso, y otros, 2004), entre ellos podemos mencionar:

- Estilos de Flujo de Datos: Filtros y Tuberías.
- Estilos Centrados en Datos: Arquitectura de Pizarra y Repositorio.
- Estilos de Llamada y Retorno: Modelo Vista Controlador, Arquitectura en Capas, Arquitectura Orientada a Objetos y Arquitectura Basada en Componentes.
- Estilo de Código Móvil: Arquitectura de Máquinas Virtuales.
- Estilos Peer to Peer: Arquitecturas Basada en Eventos, Arquitecturas Orientadas a Servicios (SOA) y Arquitecturas Basadas en recursos.

Para estructurar el diseño del simulador se decidió escoger los estilos arquitectónicos de Llamada y Retorno que se muestran a continuación pues son los que más se ajustan a la solución que se desea implementar:

Arquitectura en Capas.

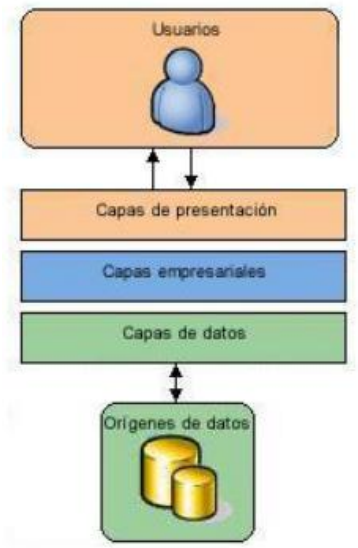


Figura 5 Arquitectura en capas.

Según (Núñez, 2008) la arquitectura en Capas es un estilo de arquitectura en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario.

El simulador utilizará arquitectura en tres capas.

Capa de presentación: es la parte visual del software que ve el usuario, y en la que podrá visualizar las notificaciones del sistema así como la entrada o cambios de datos del mismo.

Capa de negocio: es donde estarán los programas que se ejecutarán tras la petición de un usuario. Esta capa comunica la capa de presentación en la cual interactúa el usuario con la capa de datos.

Capa de datos: es donde se ubican los datos del dominio, en este caso los datos no serán almacenados en almacenes de datos o bases de datos, serán almacenados en archivos temporales creados por el sistema.

Modelo – Vista – Controlador.

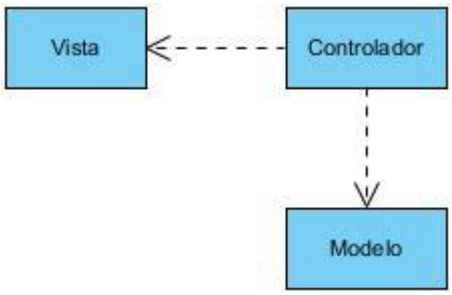


Figura 6 Modelo-Vista-Controlador.

El patrón Modelo-Vista-Controlador (MVC) separa el modelo de dominio, la vista de presentación y las acciones llevadas a cabo entre la presentación y dominio.

Modelo: administra los datos del dominio de la aplicación.

Vista: la interfaz visual que representa las funcionalidades del sistema y con las cuales interactúa el usuario.

Controlador: interpreta las peticiones del usuario para llevar a cabo acciones sobre el modelo o las vistas según sea necesario.

2.5.3 Vista lógica de arquitectura del simulador de Interbloqueos

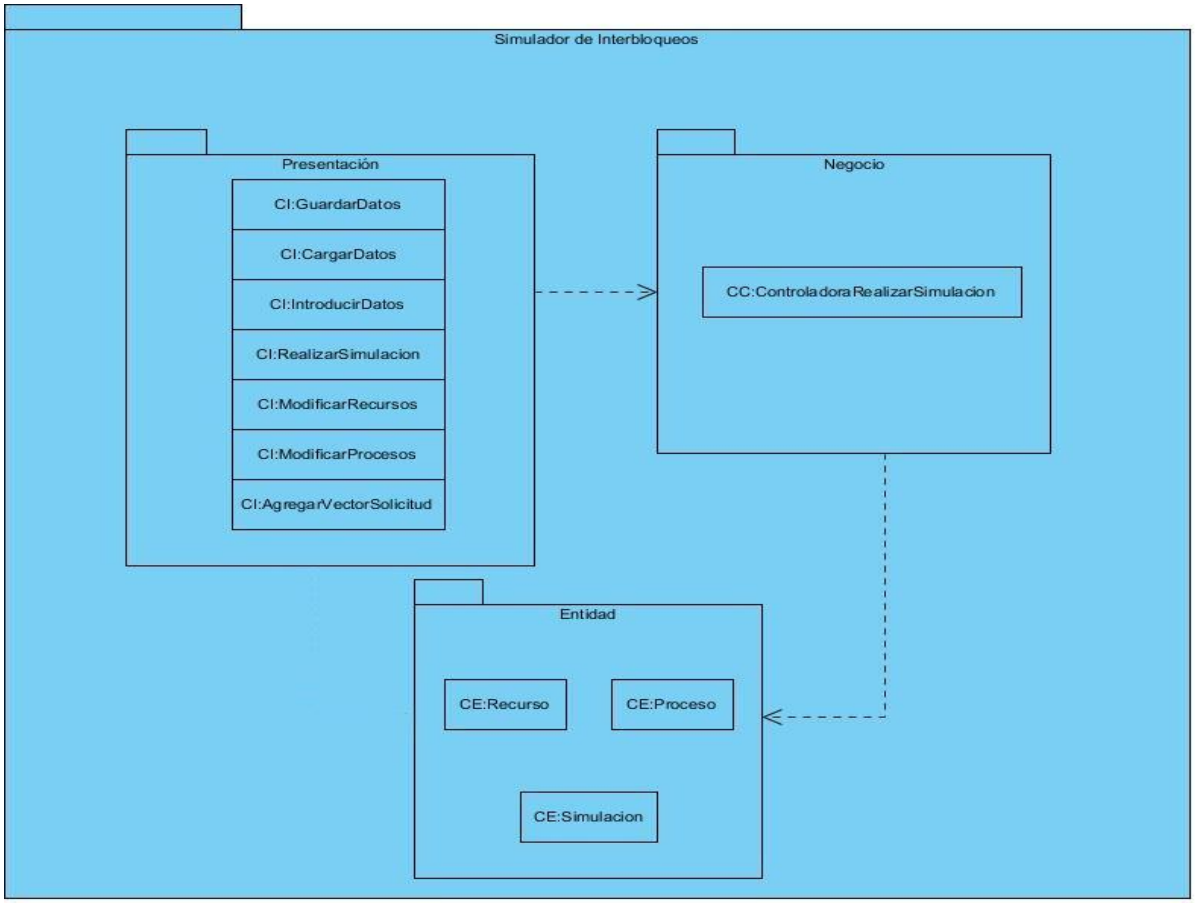


Figura 7 Vista lógica de arquitectura del simulador de Interbloqueos.

En la vista lógica se puede apreciar un paquete general llamado Simulador de Interbloqueos compuesto por tres capas (Presentación, Negocio, Entidad). Logrando de esta forma un menor acoplamiento entre las clases y respondiendo así al patrón de bajo acoplamiento.

En la capa de Presentación se encuentran todas las clases visuales necesarias, aquí se realizan las validaciones necesarias de los datos de entrada del usuario además esta capa comunica con la capa del Negocio.

En la capa de Negocio se encuentra la clase controladora que organiza el trabajo con los datos de la aplicación. En este nivel se procesan todos los datos entrados por el usuario y se le dan respuestas a los mismos. En esta capa se reciben solicitudes de la capa de Presentación y se comunica con la capa Entidad para solicitar datos y enviárselos a la capa de Presentación nuevamente.

En la capa entidad se encuentran las entidades de las clases que manejan los datos durante el funcionamiento de la aplicación. Además este nivel brinda las funcionalidades comunes a los demás niveles. Esta capa recibe peticiones de los demás niveles.

2.5.4 Diagrama de casos de uso arquitectónicamente significativos

La vista de casos de uso muestra un conjunto de casos de usos arquitectónicamente significativos del modelo de casos de uso del sistema.

En la siguiente sección se representan los casos de usos arquitectónicamente significativos así como una breve descripción de ellos ya que anteriormente se describen.

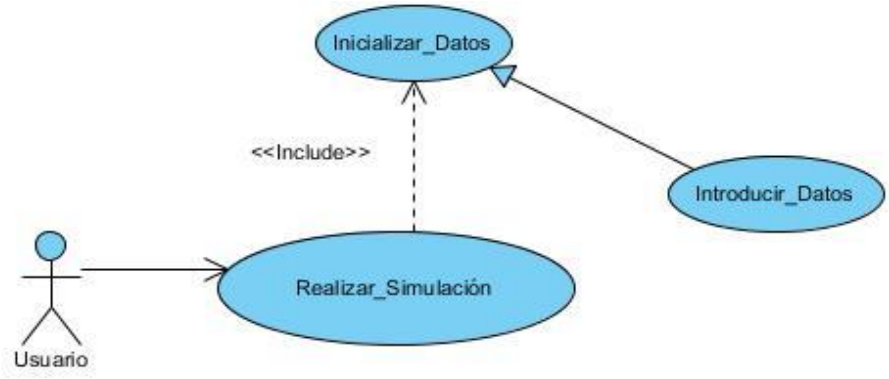


Figura 8 Diagrama de casos de uso arquitectónicamente significativos.

En su conjunto estos casos de uso forman los casos de usos arquitectónicamente significativos ya que a través de ellos queda consolidado el simulador, para realizar la simulación se necesita introducir los datos de la misma. Una vez que se esté realizando la simulación se podrá acceder a las demás funcionalidades de menos peso en la arquitectura del software.

Realizar Simulación: este caso de uso permite al usuario visualizar el proceso de simulación de interbloqueos tanto de forma algorítmica como gráfica, además puede navegar en la simulación de forma que le permita al usuario una mayor iteración con la aplicación. Además el usuario puede visualizar el estado de la simulación así como las modificaciones que trae consigo cada iteración de la simulación. Para comenzar una simulación primero se deben inicializar las variables de la misma.

Inicializar Datos: este caso de uso permite al sistema cargar los datos insertados por el usuario de

varias formas. Una vez los datos son ingresados por el usuario el sistema debe ser capaz de generar todas las variables necesarias para el funcionamiento de la simulación y dejar el ambiente listo para comenzar con la simulación.

Introducir Datos: este caso de uso es una especialización del caso base inicializar datos. Los datos son entrados por el usuario tanto manualmente como automáticamente por el propio sistema y de aquí generar las variables para iniciar la simulación.

2.6 Diseño del Simulador de Interbloqueos

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte los requisitos – incluyendo los requisitos no funcionales y otras restricciones- que se le suponen (Jacobson, Booch, & Rumbaugh, 2000).

El modelado de diseño del software es el equivalente al plano de una casa para un arquitecto. Comienza con la representación de la totalidad del objeto que será construido (por ejemplo, una reproducción tridimensional de la casa) y con lentitud lo refina para proporcionar una guía para construir cada detalle (Pressman, 2005).

2.6.1 Diagrama de clases del diseño

A través del flujo de diseño, uno de los artefactos más importantes a obtener son los diagramas de clases de diseño, donde se exponen las clases que intervienen en las realizaciones de los casos de uso del sistema. En este tipo de diagrama se representa un nivel de detalle más alto que los diagramas de clases del análisis, relacionándose con el lenguaje de programación del cual se hará uso en la implementación del sistema (Carvajal Pérez, 2009).

A continuación se muestra el diagrama de clases del diseño del simulador:

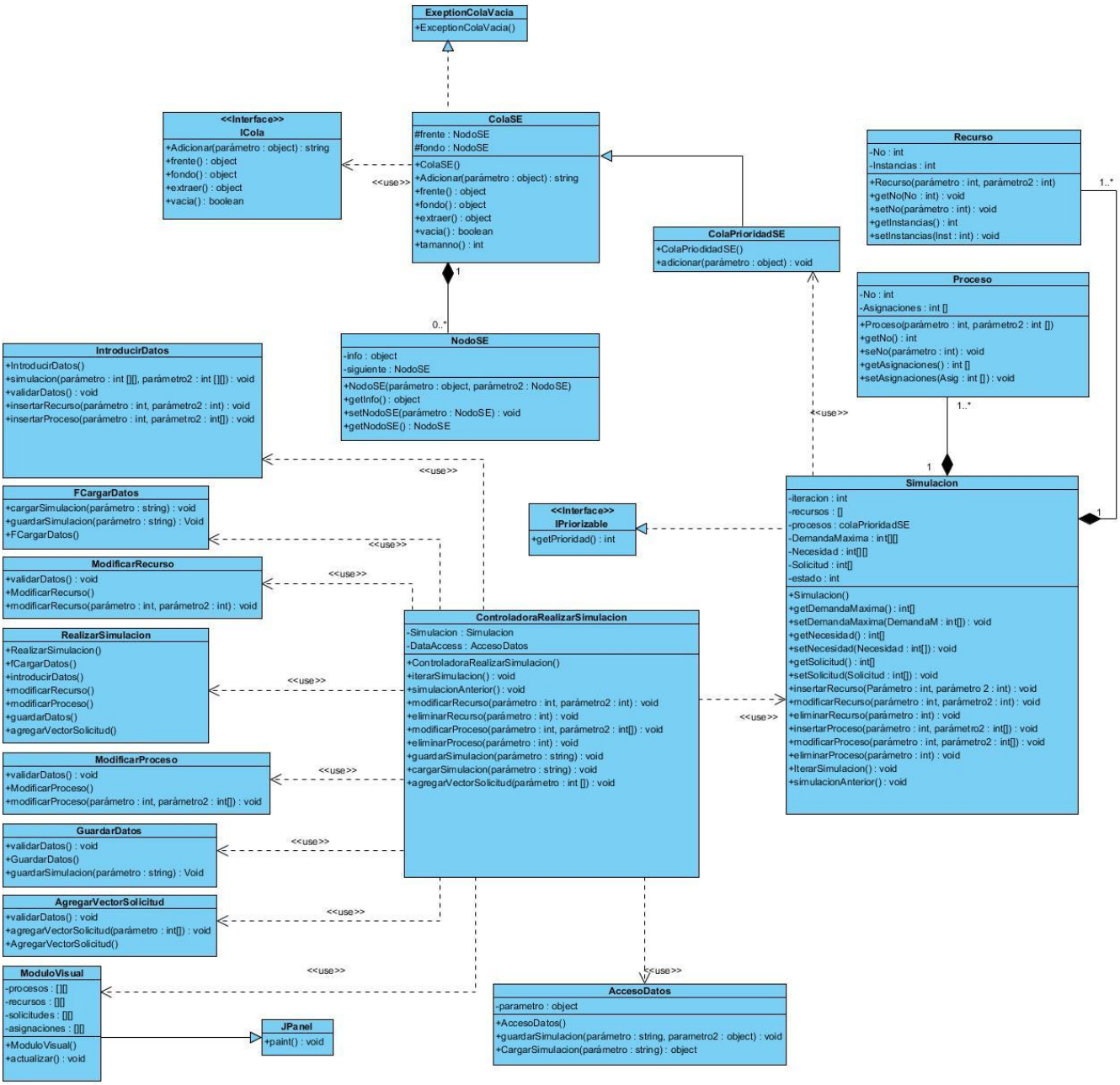


Figura 9 Diagrama de clases de diseño.

2.6.2 Patrones de diseño utilizados

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias). Los patrones son soluciones de sentido común que deberían formar parte del conocimiento de un diseñador experto. Además facilitan la comunicación entre diseñadores, pues establecen un marco de referencia (terminología, justificación) (Carvajal Pérez, 2009).

Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. Los patrones de diseño expresan un esquema organizativo estructural fundamental para sistemas de software (Bernal, 2011).

En la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas (Larman, 1999).

Patrones GRASP (General Responsibility Asignment Software Patterns) utilizados.

- **Creador:** la clase Simulación tiene la responsabilidad de crear instancias de las clases entidades pues tiene los datos de inicialización de las mismas por lo que aquí se cumple el patrón creador.

- **Controlador:** Existe una clase controladora en este caso la clase ControladoraRealizarSimulacion encargada de gestionar los eventos externos y tramitarlos las demás clases especializadas manteniendo de esta forma la relación a través de ella entre las interfaces externas y la lógica del negocio.

- **Alta cohesión:** la complejidad de las clases se distribuye entre varias clases especializadas en hacer cada una su trabajo y a la vez cada una se vuelven imprescindible para la ejecución de la simulación manteniendo así una alta cohesión entre ellas.

- **Bajo acoplamiento:** las clases son independientes y por lo tanto algunas son clases reutilizables, de esta forma se facilita el mantenimiento a cada una de ellas sin necesidad de afectar la lógica del negocio asegurando el bajo acoplamiento entre ellas.

2.6.3 Diagrama de secuencia

En este epígrafe se muestra el diagrama de secuencia Introducir datos correspondiente diseño del simulador de Interbloqueos.

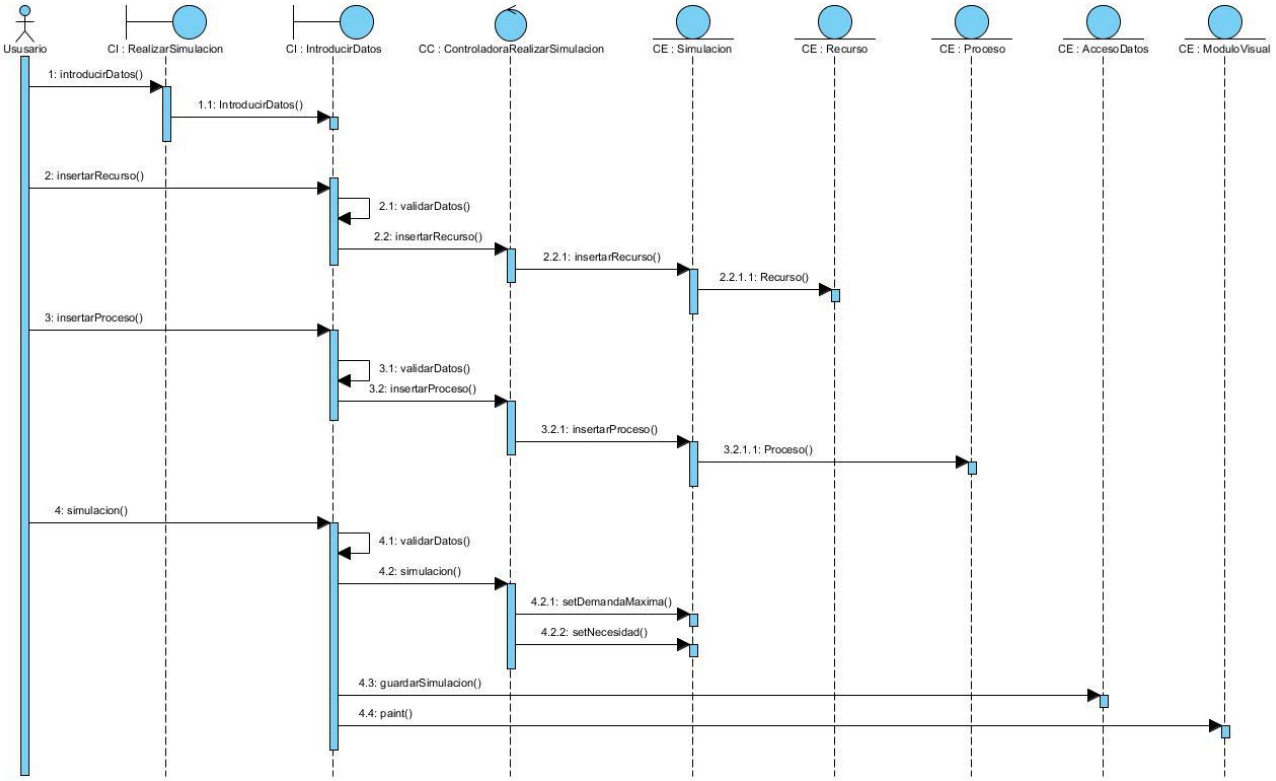


Figura 10 Diagrama de secuencia del diseño del CU Introducir Datos.

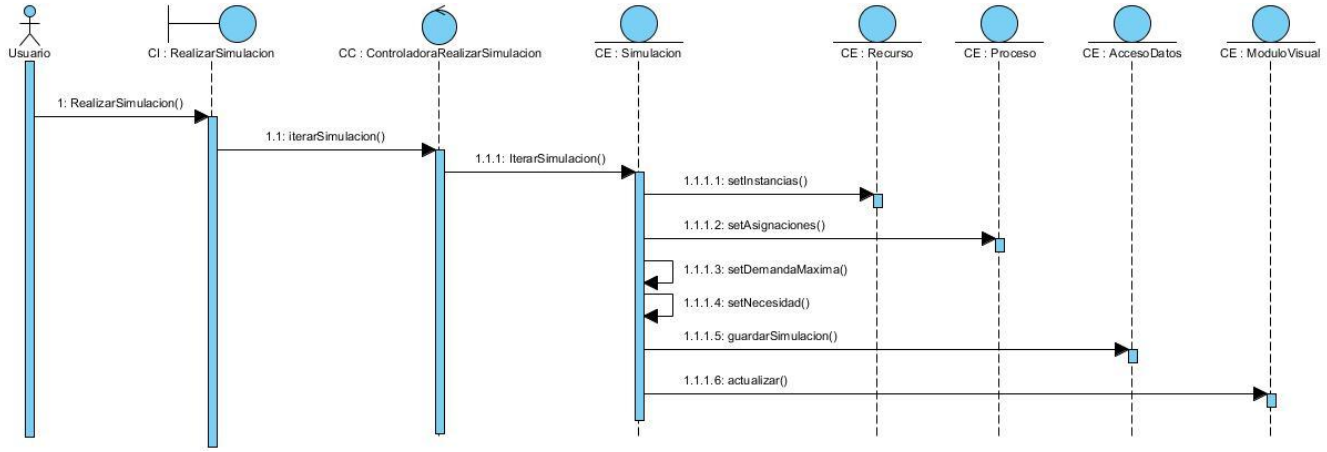


Figura 11 Diagrama de secuencia del diseño del CU Realizar Simulación.

2.7 Conclusiones

Con la realización del capítulo se obtuvieron los artefactos propuestos y que permitirán una posterior construcción del simulador. Se realizó un modelo de negocio que ayudo a comprender el funcionamiento del mismo, se capturaron los requisitos funcionales y no funcionales, fundamentales para la elaboración de los artefactos del análisis y diseño, pues a través de ellos se realizó un modelamiento preciso para luego seleccionar una arquitectura acorde con las necesidades del cliente y un diseño que garantiza la utilización de algunos patrones a fin de mejorar aspectos fundamentales, como la reutilización y la asignación de responsabilidades entre clases.

Capítulo III: Validación

En el capítulo se describen las pruebas realizadas a los diferentes artefactos generados en la captura de requisitos, el análisis y el diseño de la solución propuesta. Para realizar una correcta validación de los requisitos se utilizaron diferentes técnicas y herramientas como la lista de chequeo, la matriz de trazabilidad y el empleo de prototipos. Para validar el diseño se utilizó además la validación por tamaño de clases.

3.1 Lista de chequeos

La lista de chequeos es un documento que contiene un conjunto de parámetros medibles sobre aspectos determinados que permiten la verificación del grado de cumplimiento de los requisitos. La lista de chequeos mide y evalúa mediante un formulario con preguntas referentes a la calidad de la documentación realizada.

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Está el documento acorde con la plantilla estándar del proyecto o del expediente de proyecto?	0		0	
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)	0		0	
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**

crítico	1. ¿Están todos los requisitos redactados de forma simple y clara para aquellos que vayan a consultarlo en un futuro?	0		0	
	2. ¿Debería especificarse algún requisito con más detalle?	1		3	
	3. ¿Debería especificarse algún requisito con menos detalles?	0		0	
	4. ¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no como el sistema debe hacerlo?	0		0	
crítico	5. ¿Han sido abordadas e identificadas los valores de entradas y salidas?	0		0	
	6. ¿Han sido incluidos las respuestas válidas y no válidas de los valores de entrada?	1		1	
	7. ¿Se han identificado los requisitos de software y de hardware?	0		0	
	8. ¿Han sido identificadas				

	las restricciones de diseño e implementación?				
	9. ¿Han sido identificadas las restricciones de interfaz externa?	0		0	
	10. ¿Los requisitos de soporte y usabilidad se han identificados?	0		0	
	11. ¿Se han identificado los requisitos de seguridad (confidencialidad, integridad, disponibilidad)?	0		0	
	12. ¿Se puede verificar cada requisito? (Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requisito, ejemplo la especificación del caso de uso).	0		0	
	13. ¿Se han enumerado los requisitos incluso los que se derivan de otros requisitos?	0		0	
	14. ¿Se puede trazar	0		0	

	cada requisito al origen en el entorno del problema, (caso de uso del negocio)?				
	15. ¿Se han especificado todos los posibles cambios en los requisitos, incluyendo la probabilidad de cambio?				
	16. ¿No aparece un mismo requisito en más de un lugar del documento de especificación?	0		0	
crítico	17. ¿No existe contradicción entre lo especificado por un requisito y lo especificado por otro?	0		0	
	18. ¿Existe correspondencia entre el modelo de caso de uso, las Especificaciones Suplementarias y las especificaciones de requisitos?	0		0	
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	1. ¿Ha identificado errores ortográficos?	1		2	

Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?	0		0	
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?	0		0	
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?	0		0	

Tabla 5 Lista de chequeos de la especificación de los requisitos.

Al aplicar la lista de chequeos al documento de especificación de requisitos arrojó como resultado que en la primera iteración tres de ellos deberían describirse con más detalles, en uno de los requisitos no se había incluido las respuestas válidas o no válidas de los valores de entrada, se encontraron errores ortográficos en dos descripciones de requisitos.

3.2 Matriz de trazabilidad

La matriz de trazabilidad es un herramienta que se utiliza para la validación de requisitos funcionales, en la misma se comprueba que todos los requisitos sean cubiertos por el sistema y de esta forma detectar alguna incoherencia a tiempo.

A continuación se muestra la matriz de trazabilidad utilizada para la validación de los requisitos.

	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8
RF1	X	X	X					
RF2		X						
RF3		X						
RF4			X					
RF5				X				
RF6				X				
RF7				X				
RF8				X				
RF9				X				
RF10				X				
RF11			X		X			
RF12			X			X		
RF13							X	
RF14								X

Tabla 6 Matriz de trazabilidad.

3.3 Métricas de la calidad de la especificación

La validación de requisitos se realiza aplicando la métrica de la calidad de la especificación, examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad. Para esto es necesario conocer el total de los requisitos Rt dado por:

Rt = Rf + Rnf

26 = 14 + 12

Dónde:

Rt: Total de requisitos.
Rf: Cantidad de requisitos funcionales.
Rnf: Cantidad de requisitos no funcionales.
Los parámetros a medir son Especificidad, Consistencia Interna y Externa y Estabilidad.

- Especificidad.

Para determinar la especificidad (ausencia de ambigüedad) de los requisitos. Se explica una métrica basada en la consistencia de la interpretación de los revisores para cada requisito:
Se calcula Q1 para determinar la especificidad de los requisitos.

Alta (0.90 <= **E** <= 1).
Media (0.80 <= **E** < 0.90).
Baja (0.7 <= **E** < 0.80).

Q1 = Rui / Rt
0.92 = 24/26

Dónde:
Rui: Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.
Q1: Ausencia de ambigüedad.
Cuanto más cerca de 1 esté el valor de **Q1**, menor será la ambigüedad de la especificación.
El valor de Q1 = 0.92, esto demuestra que los requisitos se encuentran con un alto nivel de especificidad a pesar de que los RF2 y RF5 obtuvieron diferentes interpretaciones por parte de los

revisores. Para darle solución se procedió a revisarlos nuevamente para corregir la no conformidad.

- Consistencia Interna:

Esta medida se determinó con la relación Q2= (Rc-Rec)/Rc con Rc como número de requisitos especificados y Rce, número de requisitos con conflictos en la especificación. El valor de Q2 se encuentra entre 0 y 1, y el resultado más conveniente de esta métrica es el más aproximado a 1, y expresa que no existen subconjuntos de requisitos contradictorios.

Según los revisores: Rc =26 y Rce = 0;

Se sustituyen los valores:

Q2 = (Rc-Rec)/ Rc

Q2 = 26-0 /26

Q2 = 1

- Consistencia Externa:

La consistencia externa se halló con la relación Q3= Rcd /Rt con Rcd como número de requisitos consistentes en otros documentos y Rt como total de requisitos. El valor de Q3 siempre está entre 0 y 1, y su valor óptimo es el más cercano a 1, expresando que los requisitos del software no están en contradicción con los requisitos del sistema. Con Rcd =26 y Rt = 26

Se sustituyen los valores:

Q3 = Rdc/ Rt

Q3 = 26 / 26

Q3 = 1

- Estabilidad

La estabilidad fue definida con la relación $E = \frac{Rt-Rm}{Rt}$ donde Rm son los requisitos modificados, y Rt que es equivalente al número de requisitos. Se considera como valor óptimo para esta métrica el valor más próximo a 1. Luego de la sustitución correspondiente:

Se sustituyen los valores:

E = (Rt – Rm) / Rt

E = (26 – 4) / 26

E = 0.92

De manera general los valores obtenidos para cada uno de los factores, es recomendable que para que un valor sea clasificado como óptimo éste se encuentre lo más cercano a 1 posible, en este caso particular quedó establecido que cada uno de estos factores se clasificara como óptimo a partir del valor 0.84. En el caso de la estabilidad se estipuló lo siguiente:

Alta: (0.90<=E<=1);
Media: (0.80<=E<0.90)
Baja: (0.70<=E<0.80).

3.3 Métricas para validar los casos de usos del sistema

En este acápite se aplican un conjunto de métricas orientadas a objetos para evaluar las siguientes propiedades de calidad: consistencia, correctitud, completitud y complejidad. Cada uno de estos factores tendrá asociada una o más métricas, que establecen una medida cuantitativa del grado en que los factores presentan una pobre calidad. (Larman, 1999)

- **Completitud:** grado en que se ha logrado detallar todos los casos de uso relevantes.
- **Consistencia:** grado en que los casos de uso del sistema describen las interacciones adecuadas entre el usuario y el sistema.
- **Correctitud:** grado en que las interacciones actor / sistema soportan adecuadamente el proceso del negocio.
- **Complejidad:** grado de claridad en la presentación de los elementos que describen el contexto y la claridad del sistema.

Para clasificar los resultados obtenidos se establecieron las siguientes reglas:

- Alto (90% <= E <= 100%).
- Medio (80%<= E < 90%).
- Bajo (70%<= E < 80%).

Como parte de la validación de los casos de usos del sistema se decidió realizar dos revisiones, a continuación se presentan los resultados obtenidos en cada una de las revisiones:

Para una primera iteración se obtuvo un 81.25% de funcionalidad de los casos de usos del diagrama, desglosados en un 75% para la completitud, un 87% de consistencia, 100% de correctitud y un 62.5%

de complejidad, mostrando un bajo nivel en los resultados de la completitud y la complejidad por lo que se realizó una segunda iteración para elevar la funcionalidad del diagrama de casos de uso del sistema.

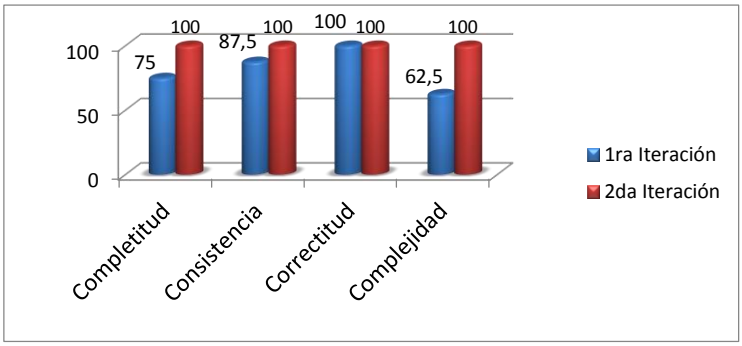


Figura 12 Gráfica de factores de la métrica para validar los casos de uso.

Como se aprecia en el gráfico anterior los resultados obtenidos en la revisión de la segunda iteración fueron relevantes y con la aplicación de estas métricas se pudieron evaluar los elementos completitud, consistencia, correctitud y complejidad del diagrama de casos de uso del sistema lo que permite observar que los requisitos identificados son abordados por todos los casos de uso y que cumplen con las reglas establecidas por las métricas. A partir de los resultados se puede comprobar que el artefacto caso de uso del sistema se encuentra con toda la calidad requerida para dar paso a la arquitectura y posterior diseño del sistema.

3.4 Métricas de tamaño de clase (TOC).

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad (Colectivo de Autores, 2001):

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

Complejidad de implementación: Consiste en el grado de dificultad que tiene que implementar un diseño de clases determinado.

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

Para medir el tamaño de las clases (TC), se tienen en cuenta los aspectos siguientes:

- Total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase.
- Cantidad de atributos (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase.
- Promedio general de los dos anteriores para el sistema completo.

Clases	No. Atributos	No. Operaciones
ControladoraRealizarSimulacion	2	9
Simulación	7	8
Proceso	2	0
Recurso	2	0
AccesoDatos	1	2
ModuloVisual	4	2
ColaPrioridadSE	2	6
ColaSE	0	7
NodoSE	2	0
ExeptionColaVacia	0	0

Tabla 7 Clases del diseño.

Para evaluar las métricas son necesarios los valores de los umbrales. Las clases se clasifican en tres grupos, según su tamaño, los que se presentan en la siguiente tabla junto con los umbrales seleccionados para su clasificación.

Umbral	Tamaño	Cantidad de Clases
<=20	Pequeño	10
>20 y <=30	Mediano	0
>30	Grande	0

Tabla 8 Umbral

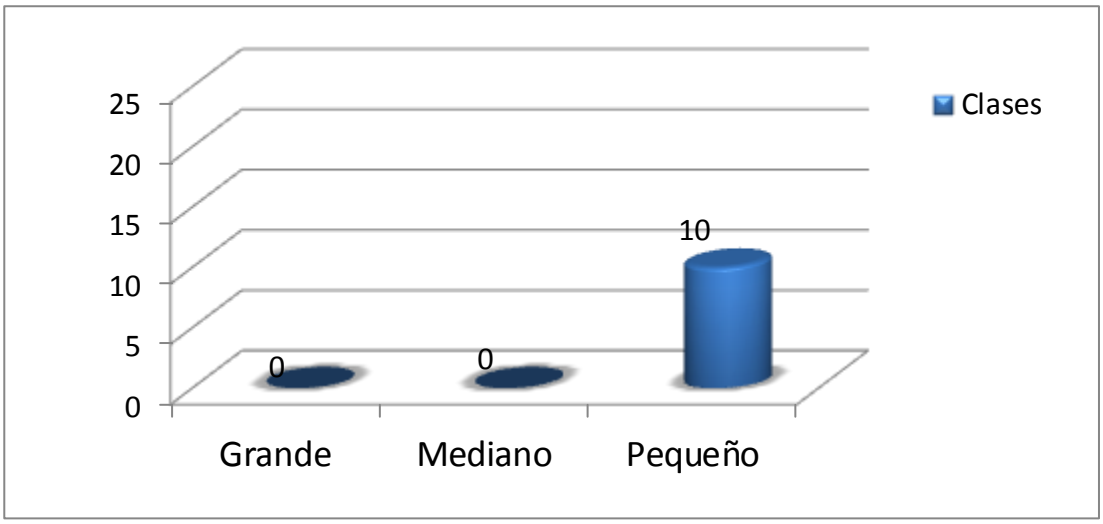


Figura 13 Resultados de la métrica de tamaño de clases.

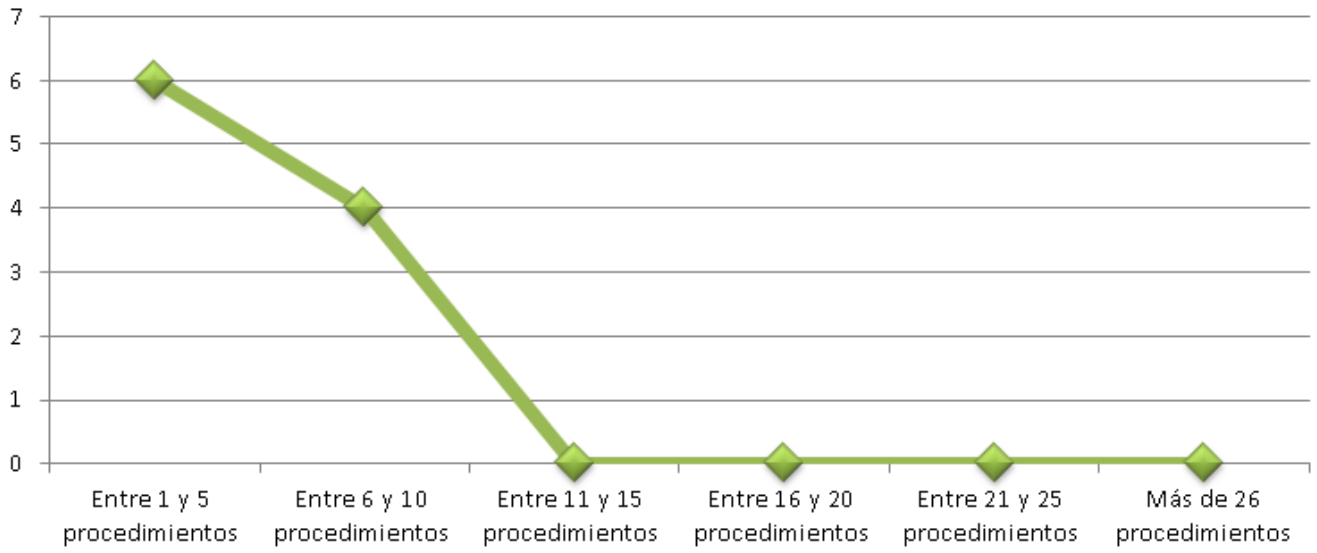


Figura 14 Representación de los resultados por intervalos definidos.

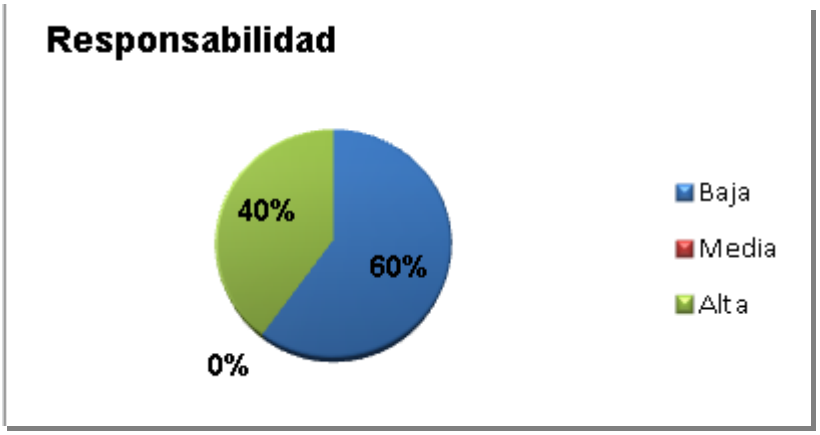


Figura 15 Resultados de la métrica TOC en el atributo Responsabilidad.

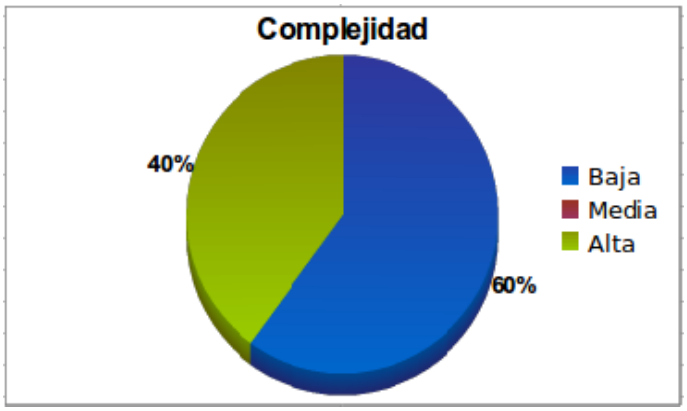


Figura 16 Resultados de la métrica TOC en el atributo Complejidad de Implementación.

Figura 17 Resultados de la métrica TOC en el atributo Reutilización.

Al utilizar esta métrica se pudo comprobar que atendiendo a la cantidad de operaciones, el 100 % de las clases diseñadas están consideradas como pequeñas, lo que facilitará el proceso de construcción del sistema. Además el sistema cuenta con un 60% de clases con poca complejidad por lo que pueden ser fácilmente implementadas y posteriormente reutilizadas con un porciento de reutilización del 60%.

3.5 Conclusiones

Con las validaciones realizadas en el capítulo se pudieron corregir aspectos fundamentales referente a la captura de requisitos y las descripciones de los mismos, factor fundamental para un correcto modelaje de un producto que debe satisfacer las necesidades del cliente. Igualmente se validaron los casos de uso del sistema, los mismos tuvieron que ser corregidos en todos sus aspectos para lograr una mayor calidad es sus especificaciones. Para validar el diseño se utilizó la métrica de tamaño de clases (TOC), con la misma se pudo comprobar que la implementación del sistema no sería de gran complejidad para los programadores pues el mismo cuenta con clases de poca complejidad y algunas de ellas reutilizables como se puedo comprobar en los gráficos mostrados anteriormente. Con las métricas de validación utilizadas se asegura la construcción de un sistema de calidad que cumple con lo especificado por el cliente.

Conclusiones

- Mediante el estudio de los interbloqueos del tema de Gestión de procesos, los laboratorios virtuales y los diferentes simuladores creados en Cuba y el mundo, se logró obtener una visión más precisa de lo que debía construir, cómo y con qué se debía construir.

- Mediante el Visual Paradigm para el modelado, utilizando RUP como metodología y UML como lenguaje de modelado se lograron obtener los artefactos propuestos en las fases de análisis y diseño con todas las ventajas que brindan RUP y UML en cuanto a flexibilidad, documentación y claridad.

- Con la elaboración de los diagramas de casos de uso y la utilización de patrones en los mismo, los diagramas de clases del análisis, la selección de la arquitectura así como sus patrones, el diagrama de clases del diseño y los diagramas de secuencia, validados correctamente por métricas y técnicas de validación con las cuales se pudo comprobar la calidad que representaban los mismos, se puede proceder a una construcción rápida y precisa del sistema.

Recomendaciones

Se recomienda:

- La implementación del Simulador de Interbloqueos para el Laboratorio Virtual de Sistemas Operativos y su posterior incorporación al Módulo de Simuladores del Laboratorio Virtual de Sistemas Operativos para que sea utilizado por los estudiantes y profesores de la asignatura.

- Mejorar iterativamente el simulador incorporándole otras funcionalidades que sean necesarias como otros contenidos del tema de Gestión de Procesos para aumentar la calidad del mismo.

Bibliografía

E. Perry, Dewayne y L. Wolf, Alexander . 1992. *Foundations for the Study of Software Architecture.* EEUU : s.n., 1992.

Rodríguez Chávez, M.Sc. Lilia Ester y Rubén Quesada, Dra. Carmen Mercedes . 2009. *La simulación computarizada como herramienta didáctica de amplias posibilidades.* Habana : s.n., 2009.

(RUP), Rational Unified Process. **computación, Departamento de Sistemas Informáticos y. 2009.** Valencia : s.n., 2009. Rational Unified Process (RUP). págs. 1-24.

Alhir, Sinan Si. 2003. *Learning UML.* EEUU : O'Reilly & Associates, Inc, 2003.

Bernal, Yuliet Legrá. 2011. Ecured.cu. *Ecured.cu.* [En línea] 19 de 3 de 2011. [Citado el: 10 de 5 de 2012.] http://www.ecured.cu/index.php/Patrones_de_diseño_y_arquitectura.

C. Suarez, Mayrilis y S. Marrero, Daymi. 2009. *Teleidentificador Personal: Diseño de la arquitectura de la plataforma manejadora de peticiones.* Habana : s.n., 2009.

Carvajal Pérez, Erlys. 2009. *Análisis y diseño del subsistema de Análisis de Resultados de un Simulador de Procesos Químicos.* Habana : s.n., 2009.

Colectivo de Autores. 2001. Metricas para el diseño de software. [En línea] 2001. [Citado el: 17 de Junio de 2012.] www.infor.uva.es/~manso/calidad/metricasoo-2011.pdf.

Figueredo Jiménez, Lianni Y. y Rodríguez Urrutia, Misael . 2010. *Simulador para la asignatura Sistemas Operativos.* Habana : s.n., 2010.

Fundora Echevarria., Leonard. 2008. *Búsqueda de los elementos arquitectónicos del software educativo en la UCI.* Habana : s.n., 2008.

García, Reynier Rivas. 2011. Ecured.cu. [En línea] 12 de 7 de 2011. [Citado el: 21 de 4 de 2012.] http://www.ecured.cu/index.php/Simulación_(Informática).

Hung, Lanyín Zaldívar. 2011. Ecured.cu. *Ecured.cu.* [En línea] 21 de 9 de 2011. [Citado el: 8 de 5 de 2012.] http://www.ecured.cu/index.php/Ingeniería_de_requisitos.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : ADISSON WESLEY, 2000.

Larman, Craig. 1999. *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* s.l. : Prentice Hall, 1999.

Levene, Ricardo y Alicia , Chiaravalloti. 1998. *Delitos Informaticos.* Argentina : s.n., 1998.

Morales, Antonio Marichal. 2011. Ecured.cu. *Ecured.cu.* [En línea] 9 de 5 de 2011. [Citado el: 3 de 5 de 2012.]

http://www.ecured.cu/index.php/CASE.

Núñez, Vladimir Milián. 2008. *Diseño de la arquitectura del proyecto SIGAC.* Habana : s.n., 2008.

Orallo, Enrique Hernández. 2002. *El lenguaje Unificado de Modelado (UML).* 2002.

Pérez, Erlys Carvajal. 2011. Ecured.cu. *Ecured.cu.* [En línea] 19 de 4 de 2011. [Citado el: 26 de 4 de 2012.] http://www.ecured.cu/index.php/Simuladores_de_Procesos.

Pressman, Roger S. 2005. *Ingeniería del Software. Un enfoque rápido. Sexta edición.* s.l. : MC Graw Hill, 2005.

Reynoso, Carlos y Kiccillof, Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

Roque, Dalida y otros, y. 2012. *Informe semestral de la asignatura de Sistemas Operativos.* Habana : Colectivo de asignatura de la Facultad 3, 2012.

Rosado, L y R. Herreros, J. 2009. *Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la física.* Portugal : s.n., 2009.

Salas Perea, Dr. Ramón S. y Ardanza Zulueta, Dr. Plácido . 1995. http://www.bvs.sld.cu. *http://www.bvs.sld.cu.* [En línea] 6 de 1995. [Citado el: 27 de 4 de 2012.] http://www.bvs.sld.cu/revistas/ems/vol9_1_95/ems03195.htm.

Sistemas Operativos - Actividad 9. Interbloqueo. **asignatura, Colectivo de Profesores de la. 2011.** Habana : s.n., 2011. Actividad 9. Interbloqueo. págs. 1-6.

Sommerville, Ian. 2005. *Ingeniería del Software. Séptima edición.* Madrid : Addison Wesley, 2005.

Stallings, William. 1997. *Sistemas Operativos.* Madrid : P R E N T I C E H A L L, 1997.

UCI, Colectivo de profesores. 2012. *Bases y principios del proceso de enseñanza-aprendizaje centrado en el aprendizaje en la UCI.* Habana : s.n., 2012.

Vary, James P. 2000. *Informe de la reunión de expertos.* Francia : s.n., 2000.