Propuesta de Validación de Modelos Conceptuales e Interfaces a Través de Modelos Abstractos

Juan Carlos Moreno [1], Marcelo Martín Marciszack [1], Mario Alberto Groppo [2]
Centro de Investigación, Desarrollo y Transferencia de Sistemas de Información (C.I.D.S)
Universidad Tecnológica Nacional – Facultad Regional Córdoba
Maestro Lopez esq. Cruz Roja Argentina, Ciudad Universitaria, Córdoba, República Argentina
{jmoreno33, marciszack}@gmail.com [1], sistemas@groppo.com.ar [2]

Abstract

El presente trabajo tiene como objetivo presentar un procedimiento para la validación de Modelos Conceptuales, brindando como ventaja adicional la posibilidad de verificar algunos atributos de Usabilidad y la Accesibilidad entre interfaces de un Sistema de Información. Para dicho objetivo, se emplean modelos derivados que utilizan una variante del lenguaje de intercambio XML, para validarlos a través del uso de modelos abstractos. La propuesta parte del modelo de negocio, donde captura las actividades a partir de las cuales generará los Casos de Uso y el diccionario de datos. Una vez terminado de especificar los mismos, se construyen los prototipos de las interfaces y empleando el paradigma del Desarrollo de Software Dirigido por Modelos, mediante transformaciones se obtienen modelos abstractos que pueden ser validados con técnicas de grafos. Se brinda, además, la posibilidad de hacer correcciones y simular cambios a través de la trazabilidad de los modelos.

1. Introducción.

La industria del software actual le exige a la Ingeniería de Sistemas el desarrollo de métodos para la construcción veloz y correcta de aplicaciones de sistemas de información, que permitan aprovechar los costos de oportunidad y asegurar la subsistencia de las compañías en un mercado tan competitivo como el actual. Sobre todo, porque los costos de oportunidad son muy altos. Esto se ha acentuado más en la última década, con el desarrollo de las comunicaciones, la nanotecnología y el avance de Internet. Todas estas cuestiones, son las que motivaron el desarrollo de la siguiente propuesta, que se expone a continuación.

En el dominio de modelado de la Ingeniería de Software, existen variadas metodologías para modelar el negocio y el sistema de información. La presente propuesta parte del modelo de negocios, capturando los requisitos, para lo cual emplea BPMN, y luego a través de un proceso de transformación se construyen los casos de uso o escenarios que se modelan. Una vez que se ha concluido la construcción y modelado los distintos casos de uso, conjuntamente con la definición del diccionario de datos, se diseña un primer prototipo de modelo de la interfaz de usuarios. A partir de esta etapa, se emplea un proceso de transformación, mediante el cual se convierte la interfaz en un modelo de objetos, aplicando el paradigma de desarrollo de software dirigido por modelos. Con el modelo construido, y conociendo los requerimientos funcionales del sistema de información, se aplica un método de validación a los fines de comprobar la accesibilidad de las distintas interfaces. Además, brinda la posibilidad de simular la ejecución de tareas, comprobando la completitud de las interfaces y adicionalmente detectar la presencia de errores. En caso, de ser necesaria una modificación o corrección de la misma, es factible analizar el impacto del cambio aplicando un proceso de seguimiento de la trazabilidad de las tareas de cada interfaz analizada.

Para explicar el proceso de modelado propuesto, el documento se organiza en secciones de la siguiente manera:

En la sección 2, se hace una breve introducción al Estado del Arte, introduciendo los principales conceptos tenidos en cuenta que dieron origen en el presente trabajo.

En la sección 3, denominada Elementos de Trabajo y Metodología, en una primera parte, se presentan los conceptos y bases teóricas sobre los que se sustenta la propuesta metodológica descripto. En la misma se presentan conceptos relacionados con el Modelado Conceptual, el Desarrollo de Software Dirigido por Modelos, y el modelado de la interfaz de usuario, a través de modelos abstractos. Asimismo, se describe el procedimiento completo para validar y verificar las interfaces construidas empleando modelos abstractos. En la segunda parte, de la sección 3, se describen distintas herramientas que materializan la propuesta del procedimiento metodológico descripto.

Finalmente, en la sección 4 se presentan las conclusiones, sobre la propuesta desarrollada.

2. Estado del Arte

Uno de los objetivos de la Ingeniería de Sistemas es construir aplicaciones de calidad, útiles a los usuarios finales, aplicando distintos métodos y principios [1]. La calidad de las aplicaciones web se ha medido generalmente basándose muchas veces en el sentido común de los desarrolladores [2].

Por lo general, en el proceso de construcción del software se hace énfasis en los aspectos de la arquitectura, la funcionalidad y la persistencia de cada proceso, no tratándose de forma adecuada la interacción y facilidad de uso. Por este motivo, el estudio de la usabilidad del software en entornos web ha tomado relevancia.

La norma ISO/IEC 9126-1 [3], se considera a la usabilidad como un parámetro de calidad del software. Se reconoce a la usabilidad como "la capacidad en que un producto de software puede ser entendido, aprendido y usado por determinados usuarios bajo ciertas condiciones en un contexto de uso específico". Se contempla la calidad interna, externa y en uso de un producto de software [4]. A su vez, la usabilidad es descompuesta en sub-atributos, haciendo que algunos atributos sean más tangibles y se puedan medir [5].

La norma ISO/IEC 25000 (SQUARE) [6], contempla a la usabilidad bajo dos puntos de vista distintos: uno que contempla a la usabilidad desde el punto de vista del software, como producto en sí mismo; y el otro punto de vista desde la usabilidad de uso, desde la perspectiva del usuario.

Pero la usabilidad es considerada en etapas finales de la construcción del software, cuando cualquier modificación afecta la arquitectura del sistema y el costo de cualquier modificación es alto [7], [8]. Una de las soluciones posibles a este problema, es incluir el análisis de la usabilidad en etapas tempranas, durante la fase de elicitación de los requisitos. Por esta razón se estudia el Entorno de Desarrollo de Software Dirigido por Modelos (DSDM) [9][10], también denominado MDD en el campo de la Ingeniería de Software, puesto que se busca saber si se considera la elicitación de requisitos de usabilidad en etapas de desarrollo tempranas de la construcción del software. En DSDM se busca la construcción de un software a través de una serie de modelos conceptuales que son independientes de la plataforma de implementación y representan del sistema de información. A través de estos modelos se busca generar el código final del programa, aplicando una serie de transformaciones.

El proceso de transformación involucra cuatro niveles (ver Figura 1), que están compuestos por una serie de modelos conceptuales:

• **Modelo Independiente de Computación (CIM)**: en esta etapa la representación de los requisitos y del entorno es independiente de cualquier soporte de computación. Se lo denomina modelo de dominio, y para su construcción se utiliza vocabulario familiar a los expertos del dominio, sin que éstos sepan o tengan conocimientos técnicos de los artefactos que se utilizarán en la implementación del sistema.

• **Modelo Independiente de Plataforma (PIM)**: Es un modelo de alto nivel de abstracción (modelo conceptual) independiente de cualquier tecnología o lenguaje de implementación. Puede ser implementado en cualquier plataforma específica.

• **Modelo Específico de Plataforma (PSM)**: un PIM se transforma en uno o varios PSM. El PSM representa al PIM en una tecnología de implementación específica.

• **Código:** Es la transformación de cada PSM a código expresado en un lenguaje de programación específica para la transformación del sistema.

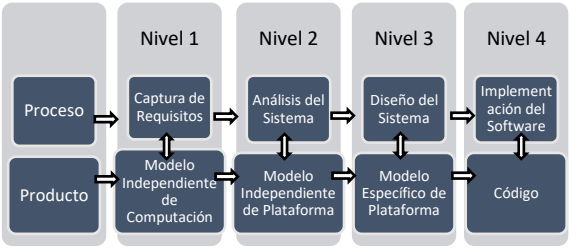

Figura 1. Esquema de Proceso de Model-Driven Architecture

Como se observa en la Figura 1, la idea fundamental de MDD es la generación automática de código a través de transformaciones automatizadas aplicadas a modelos que deben ser completos y consistentes. Se observa como cada etapa del ciclo de vida de desarrollo del software se vincula con los distintos modelos de las distintas etapas del paradigma de Desarrollo de Software Dirigido por Modelos. Aplicando el paradigma del Desarrollo de Software Dirigido por modelos, donde se tienen en cuenta los distintos niveles del ciclo de vida de construcción del software, la usabilidad puede ser implementada como un modelo que se aplica en los distintos niveles de abstracción del paradigma de MDD. Por ejemplo: 1) a nivel de Modelo Independiente de Plataforma (PIM), se aplica analizando y evaluando distintos modelos como: el modelo de navegación, los modelos de representación de interfaz abstracta del usuario o distintos modelos estructurales del dominio bajo estudio; 2) a nivel de modelos específicos de plataforma (PSM), se aplica evaluando los distintos modelos de interfaz concreto; 3) a nivel de aplicación web o de implementación final (CM), evaluando la interfaz de usuario final; y a nivel de interacción, se aplica cuando la aplicación web está siendo empleada en un determinado contexto. La presente propuesta introduce y evalúa la presencia de usabilidad durante la construcción del modelo conceptual hasta la finalización del mismo, iniciando desde la etapa de

elicitación de requerimientos del sistema en el modelo de negocios.

Existen varios métodos de desarrollo de software de la Ingeniería web, que dan soporte al estándar MDD. Se pueden citar, como ejemplo, a los siguientes: OOHDM [11] [12], UWE [13] [14], OO-Method [15], OOH[16], OOWS[17], WebML[18].

El desarrollo de los sistemas Web en estos métodos, se lleva a cabo mediante modelos que capturan distintas vistas del sistema: un modelo estructural (modela contenido y comportamiento), un modelo de navegación (modela acceso al contenido) y un modelo de presentación abstracto (modela cómo el contenido es mostrado). Se debe considerar el nivel de abstracción de los modelos para poder evaluar las características de usabilidad.

Como consecuencia del Análisis del Modelo Conceptual de distintos modelos de diseño de software web, se llevó a cabo esta investigación donde surgió esta propuesta de trabajo con el fin de incorporar aspectos de usabilidad en forma temprana en el diseño de aplicaciones de entorno web.

3. Elementos de Trabajo y Metodología.

3.1. Pasos del Proceso Metodológico.

La metodología que aquí se presenta, se conforma de las siguientes etapas, explicadas con mayor detalle a lo largo de este documento.

1) Realizar el modelado de negocio con “Business Process Model and Notation” (BPMN) o lo que sería lo mismo en español “Modelo y Notación de Procesos de Negocio” [19].

2) Indicar qué actividades están vinculadas con la gestión de información. Incorporar estereotipos (Reglas de Negocio) a aquellas actividades automatizadas con Requerimientos No Funcionales (R.N.F.) de Usabilidad.

3) Construir los Casos de Usos a partir de las actividades y el diccionario de datos empleando Léxico Extendido del Lenguaje.

4) Trasformar los Procesos de Negocios y Casos de Uso o Escenarios en máquinas abstractas.

5) Verificar la consistencia de los requerimientos y de las definiciones de atributos de Usabilidad a través de Redes de Petri Coloreadas y Autómatas Finitos resultantes del proceso anterior.

3.2. Descripción del Proceso Metodológico

3.2.1 Realizar el Modelado de Negocio con BPMN.

En BPMN, los Procesos de Negocio involucran la captura de una secuencia ordenada de las actividades e información que utiliza el proceso, el cual implica representar cómo una empresa realiza sus objetivos centrales. BPMN es una notación basada en diagramas de flujo para definir procesos de negocio, desde los más simples hasta los más complejos y sofisticados, para dar soporte a la ejecución de procesos. BPMN es gráficamente más rico, con menos símbolos fundamentales, pero con más variaciones de éstos, lo que facilita su comprensión por parte de gente no experta. A continuación, en la Figura 2 se visualiza un proceso de negocio.

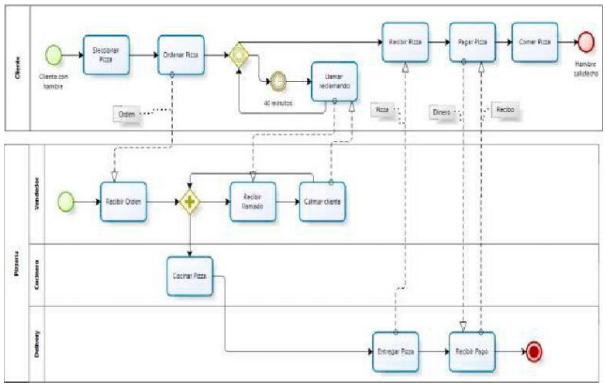

Figura 2. Ejemplo de un Modelo de Procesos de Negocio modelado con BPMN.

3.2.2 Indicar qué actividades tienen manejo de información.

En la Figura 3, se muestra cómo se procede para seleccionar las actividades de negocio que son automatizadas y formarán parte del Sistema de Información

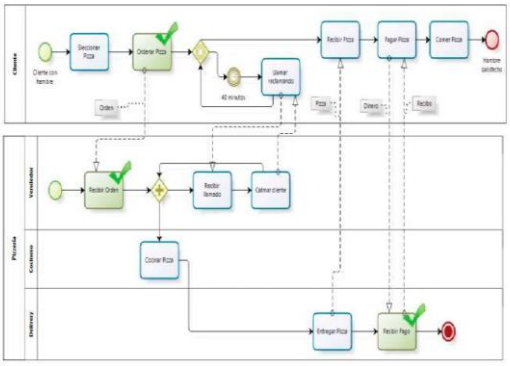

Figura 3. Selección de actividades automatizadas

Aquí, el analista deberá identificar en los diagramas de procesos aquellas actividades que utilicen / generen

información, diferenciándolas de aquellas que son puramente manuales.

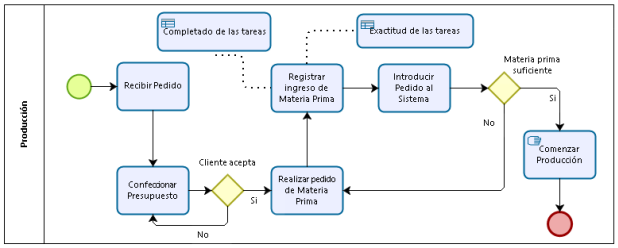

Figura 4. Modelado de atributos de usabilidad en BPMN para actividades de proceso de producción.

Las especificaciones de usabilidad deberán agregarse como actividades utilizando el estereotipo “Regla de Negocio”. En la Figura 4 se muestra un ejemplo de la actividad “Registrar ingreso de materia prima”, a la que se le asociaron los atributos “Completado de las tareas” y “Exactitud de las tareas” correspondientes a la sub-característica “Rendimiento de las tareas del usuario”.

3.2.3 Construir Casos de Uso a partir de las actividades.

Un Caso de Uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. En el contexto de la ingeniería de software, un Caso de Uso es una secuencia de acciones que se desarrollarán entre un sistema y sus usuarios en respuesta a un evento sobre el propio sistema.

Los Diagramas de Casos de Uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los Casos de Uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo, la especialización y la generalización son relaciones.

Como técnica de extracción de requerimiento un Diagrama de Casos de Uso permite que el analista se centre en las necesidades del usuario, qué espera éste lograr al utilizar el sistema, evitando que la gente especializada en informática dirija la funcionalidad del nuevo sistema basándose solamente en criterios tecnológicos. A su vez, durante la extracción (elicitation en inglés), el analista se concentra en las tareas centrales del usuario describiendo por lo tanto los Casos de Uso que mayor valor aportan al negocio. Esto facilita luego la priorización de los requerimientos. Los Casos de Uso evitan típicamente la jerga técnica, prefiriendo el léxico del usuario final.

Desde una perspectiva tradicional de la ingeniería de software, un Caso de Uso describe una característica del sistema.

En este paso, utilizando como guía las actividades marcadas como no manuales en el punto anterior, es preciso identificar los Casos de Uso del sistema que darían soporte "informático" a las actividades.

3.2.4 Trasformar los Procesos de Negocios y Escenarios en Autómata Finito.

Las máquinas abstractas son usadas para modelar una gran variedad de sistemas en diversas áreas. En este caso utilizamos un tipo particular de máquina abstracta que son los Autómatas Finitos.

Entonces, una vez que se cuenta con una especificación detallada de los Procesos de Negocios y Casos de Usos que satisfacen las necesidades informáticas del negocio, se realiza la trasformación de los mismos a máquinas abstractas.

Para formalizar esta transformación se definieron un conjunto de reglas de conversión bidireccionales:

Partiendo de BPMN: Cada una de las actividades de Negocio identificadas tendrá un mapeo directo con cada estado identificado del Autómata finito. Lo mismo ocurrirá con los estados de Inicio y de finalización, ya sea por el éxito del procedimiento o por el fracaso del mismo. Los arcos del autómata finito surgirán a través de los flujos de trabajo que vinculan las Actividades del proceso de Negocio.

Partiendo de Plantilla de Caso de Uso: El estado inicial lo constituye las precondiciones, cada acción o respuesta del sistema son arcos del AF, siendo los episodios alternativos las transiciones a estados diferentes con diferentes entradas. Este proceso es soportado tanto para CU a nivel de trazo Grueso y Finos, lo único que varía es la cantidad de estados que la constituyen.

3.2.5 Verificar la consistencia de las definiciones a través de los Autómatas Finitos resultantes.

Al contar con la representación de cada Proceso de negocio o Caso de Uso, sobre cada uno de los mismos se obtiene un Autómata Finito, a partir de una transformación directa de la representación del modelo que debe ser validado, podemos efectuar distintas acciones:

- **Conjunto Conexo y accesibilidad de estados.**

Estas verificaciones resultan fundamentales para verificar que todas y cada una de las abstracciones de estados por los que transita el AF tienen correlación con el planteo del mismo, ya que, si un estado definido en el AF no es accesible desde el estado inicial, significa que el modelo que está siendo representado por el autómata no está correctamente planteado y debe necesariamente ser reformulado.

Si un autómata no es conexo basta con eliminar los estados inaccesibles (estados no conexos) y todas sus transiciones (las de entrada y las de salida) para obtener un nuevo autómata conexo equivalente.

- **Autómata Finito Determinista.**

La forma de definir los modelos de procesos puede resultar en caminos o procesos paralelos o simultáneos, los que se traducen en no determinismo dentro de los Autómatas Finitos, los cuales merecen una especial atención de su conveniencia en mantenerlos en los modelos. Así, es necesario convertir el AF No Determinista en uno Determinista equivalente, de manera de brindar al analista la posibilidad de analizar si se reformula el modelo o se mantiene tal como está definido.

- **Minimización del Autómata Finito.**

Un AF no mínimo significa la presencia de estados equivalentes, los cuales pueden ser identificados y reemplazados, y de esta manera simplificar el Modelo que representa al Proceso de Negocio (en el proceso de Negocio dos estados equivalentes del AF equivalen a la existencia de una reinvocación de una acción que puede ser eliminada).

- **Simulación de Ejecución de Autómatas Finitos.**

Para cada modelo de proceso de Negocio y su correspondiente representación del Autómata Finito, pueden establecerse un conjunto de entradas, que al ser simuladas, verifican si se producen los resultados esperados por el modelo.

Luego de hacer esto podemos realizar una trazabilidad inversa hacia los procesos y determinar si en el proceso hay actividades que no se realizan (a partir de los estados no conexos) y procesos que son irrelevantes o innecesarios desde la minimización del autómata relacionado.

3.2.6 Verificar la consistencia de las definiciones de las interfaces a través de Redes de Petri Coloreadas.

Se realiza una breve descripción del método empleado para modelar interfaces, mediante grafos. Para ello, se emplearán Las redes de Petri, que se las puede definir como un lenguaje gráfico, con un alto nivel de formalidad para la descripción y simulación de procesos. Se emplean generalmente para modelar sistemas distribuidos y concurrentes. Estos modelos son muy empleadas para el análisis de sistemas dinámicos discretos.

Su representación gráfica, está conformada por dos tipos de nodos: círculos (lugares o places) y barras (transiciones). Los lugares y las transiciones se pueden conectar por medio de arcos. Los lugares conectados con un arco a una transición son sus lugares de entrada. Los lugares, que reciben un arco de una determinada transición, son sus lugares de salida. Simular a través de una red de Petri significa el disparo de transiciones habilitadas. Se dice que una transición está habilitada cuando existe al menos un token en todos sus lugares de entrada. Los lugares contienen cero o más tokens. Cuando se dispara una transición, un token es eliminado de cada lugar de entrada, y se añade uno a cada lugar de salida. El token define el estado de cada lugar y se denomina como marcado de red al número de tokens de cada uno de los lugares.

La parte izquierda de la Figura 5 muestra un modelo con dos tareas (Tarea-1 y Tarea-2); En una Transición Iniciada o con proceso iniciado, las tareas se pueden ejecutar en cualquier orden. Una vez finalizado el proceso o Transición Finalizado, todo el proceso se repite nuevamente. Hacia la izquierda se muestra cómo el disparo de la transición "Iniciado" ocasiona que el token sea eliminado del lugar "Inicio" y se generen dos tokens en Tarea-1 y Tarea-2 (ver parte derecha de la Figura 5).

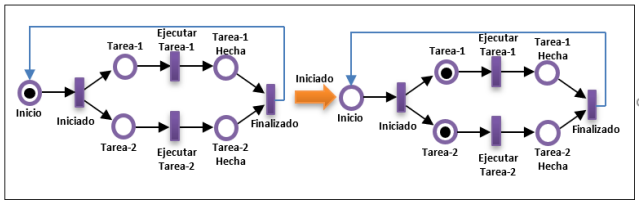

Figura 5. Simulación de una transición.

En la Figura 6 se muestra el denominado "Grafo de Alcanzabilidad", con todos los posibles estados de la red.

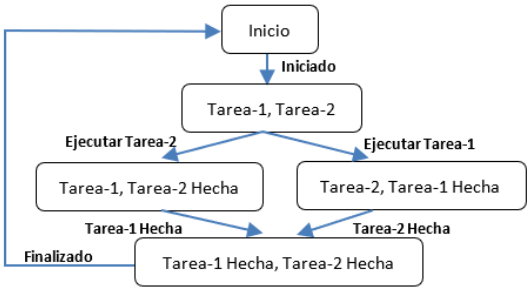

Figura 6. Grafo de Alcanzabilidad

El análisis del comportamiento de los estados del grafo, empleando simulación, permite saber si es posible alcanzar una configuración a partir de otra (alcanzar cada estado del grafo), los estados que permiten alcanzar a otros, y si existen estados inaccesibles o inalcanzables dentro de la red diseñada.

Las redes de Petri coloreadas tienen la particularidad que permiten agregar datos estructurados a los tokens, cuyo valor es a lo que se le denomina color. El mismo es importante porque permite que pueda ser seleccionado por los arcos, evaluado en las transiciones, y cambiado por transición a otro estado. Además, brinda la ventaja de

poder evaluar subredes, pertenecientes a diseños de redes mayores.

Estos modelos de redes serán empleados para validar las interfaces o pantallas de modelos abstractos.

3.3. Descripción de la herramienta

3.3.1. Construir Casos de Uso a partir de las actividades.

Esta herramienta, desarrollada específicamente para el diseño de modelos conceptuales (denominada SIAR), es una aplicación web que permite registrar en forma normalizada los casos de uso que comprende:

- Administración de los atributos de un proyecto (de sistemas) y sus versiones.
- Gestión de los alcances de cada versión del proyecto y los casos de uso asignados.
- Administración de los artefactos de un caso de uso, incluyendo actores, precondiciones, postcondiciones, escenario principal y escenarios alternativos, y su versionado.
- Clasificación, priorización y trazabilidad de los casos de uso.
- Visualización de consultas y generación de reportes en distintos formatos, inclusive XPDL, para comunicarse con otras aplicaciones.
- Gestión de atributos de procesos de negocio, de actividades de negocio que los componen y los casos de uso asociados a estas actividades.

3.3.2. Trasformar un caso de uso en una máquina abstracta.

Una vez completa la versión de un caso de uso y utilizando el conjunto de reglas de conversión del caso de uso en un grafo de estados, definidas en este paso de la metodología, la herramienta genera el grafo de estados.

El grafo de estados asociado al caso de uso tiene un alfabeto de tres símbolos para indicar qué evento lo cambia de un estado/nodo a otro:

- Por medio de una Acción determinada.
- S = Cuando Si se cumple una condición que bifurca a un escenario alternativo.
- N = Cuando No se cumple una condición que bifurca a un escenario alternativo.

Partiendo de un estado/nodo origen, en la función de transición puede estar asociado solamente uno de los símbolos: A, N o S. Con esto se cumple la condición necesaria de un autómata finito determinista. De esta manera, si la transición entre dos estados/nodos se da dentro de un mismo camino, se asocia el símbolo A. Si en cambio interviene una bifurcación, la función de transición hacia el estado/nodo destino por cumplimiento de la condición de bifurcación, se asocia el símbolo S. Por el otro camino de la bifurcación, se asocia el símbolo N. Los estados y sus relaciones (arcos pueden verse en la próxima Figura 7.

Tabla De Estados Casos de Uso: 1 - 1 - 1 - 1 CONSULTAR CURSOS Versión: 6

Estado / Paso Origen	Estado / Paso Destino	Transición	Estado Final por Error	Tipo
1	2	A		S
2	3	A		S
3	4	A		S
4	4-A	N		C
4-A	4-A-1	A		S
4-A-1	4-A-2	A	SI	S
4	5	S		C
5	6	A		S
6	6-A	S		C
6-A	6-A-1	A		S
6	7	N		C

Figura 7. Tabla de estados de un caso de uso.

3.3.3. Transformación automática.

Una vez generado el grafo de estados se expresa en protocolo XPDL, por ser el más adecuado para intercambiar modelos de procesos entre distintas herramientas. Este lenguaje da soporte a la definición y a la importación/exportación de procesos, con el objetivo de que, aunque se modele un proceso en una aplicación, este modelo pueda ser usado por otras aplicaciones de modelado y/o por otra aplicación es que trabajen en el entorno de ejecución.

A continuación, en la Figura 8 se muestra el resultado de la transformación a través de la configuración de una herramienta XLST a XML aceptado por el módulo Validador de Autómatas Finitos.

```
<transicion
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="java:dominio.automatas.TransicionFinita">
    <simbolo>
        <simbolo>S</simbolo>
    </simbolo>
    <estado-inicio>
        <denominacion>4</denominacion>
    </estado-inicio>
    <estado-fin>
        <denominacion>5</denominacion>
    </estado-fin>
    <denominacion>f( 4, S) = 5</denominacion>
</transicion>
```

Figura 8. Fragmento de archivo XML generado.

3.3.4. Simulador de Autómata Finito.

A partir de este punto es posible simular el comportamiento que tendrá el sistema y llevar a cabo la última etapa del proceso metodológico que es verificar la consistencia secuencial de los escenarios de los procesos de negocios y de los casos de uso.

Cabe en este punto, hacer la siguiente aclaración conceptual, que diferencia al concepto de modelo del concepto de simulación.

Un modelo es una representación abstracta, conceptual, gráfica, física o matemática cuyo único propósito es el de analizar, explicar, simular y describir procesos. En concreto, se realiza una abstracción del sistema mediante sus datos de entrada, para encontrar una solución a una respuesta específica.

Mientras que una simulación, es el proceso de diseñar un modelo de un sistema real, el cual se ejecutará función del tiempo. El objetivo del mismo es obtener u adquirir conocimiento de los procesos y fenómenos que ocurren, para aprender a describir y predecir su comportamiento.

La diferencia entre ambos, es que: un modelo es la abstracción de un sistema, construyéndose únicamente lo que es de interés para la solución del problema por lo cual fue creado. En cambio, la simulación es la imitación del comportamiento de un sistema a través del tiempo, con el objetivo de predecir y describir su posterior funcionamiento o comportamiento, ante determinadas entradas.

Entonces, a partir de lo anteriormente expresado, se emplean modelos para simular el comportamiento de las interfaces a partir de distintas entradas. Lo mismo ocurre con la validación de los requerimientos funcionales del sistema bajo estudio. Se emplean distintas entradas para simular el comportamiento del sistema bajo estudio de un dominio específico.

Lo que se hace es ingresar el archivo XML (salida transformada de BPMN), que representa al proceso de Negocio o caso de uso como grafo de estados, al sistema de validación de Máquinas Abstractas "Autómata Finito" (ver Figura 9).

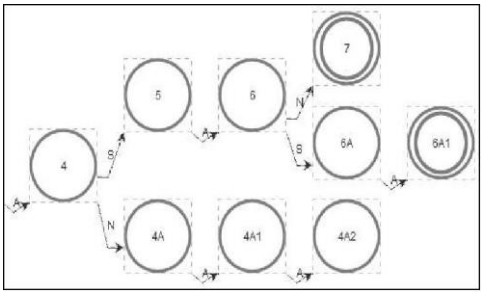

Figura 9. Grafo de estados en el simulador de autómatas finitos

3.3.5. Validaciones sobre el Autómata finito generado.

El conjunto de validaciones que se efectúan, sobre cada uno de los AF construidos a partir de los CU Derivados y de Soporte son los siguientes:

- Visualización de Autómatas Finitos en la Herramienta Validación: Definición Formal, Grafo y Tabla de Estados/Entradas
- Conjunto Conexo - Accesibilidad de Estados: Se verificará si todos los estados definidos son accesibles desde el estado inicial, con lo cual garantiza una correctitud en la definición de los mismos.
- Construcción de Autómata Finito Determinista: Para cada uno de los Autómatas Finitos construidos se informará si el mismo en caso de ser No determinista (Procesos en Parálelo) un AF Determinista equivalente, con procesos secuenciales.
- Minimización del Autómata Finito Determinista: Al Igual que el punto anterior, para cada uno de los Autómatas Finitos construidos se informará si el mismo es factible de ser minimizado, esto es hay estados equivalentes y la herramienta propone una nueva solución.
- Simulación de Ejecución: Este proceso resulta imprescindible para validar si los Procesos de Negocios representados a través del Autómata Finito respectivos están construidos de manera correcta. Este punto, se necesita una ejemplificación sobre su aplicación, ya que hay realizar una interpretación sobre las acciones descriptas en los casos de uso, y las transiciones entre estados de los AF.

El simulador posibilita probar el grafo de estados y comprobar si es aceptado o rechazado. Si es aceptado, significa que la consistencia de los escenarios del caso de uso es correcta. Si no, el rechazo, indica que habrá que revisar e introducir modificaciones en el modelo de origen al AF.

A continuación, en la Figura 10 se muestra una pantalla de salida del módulo de la herramienta de Validación en donde se muestra la secuencia de entradas y la transición de estados.

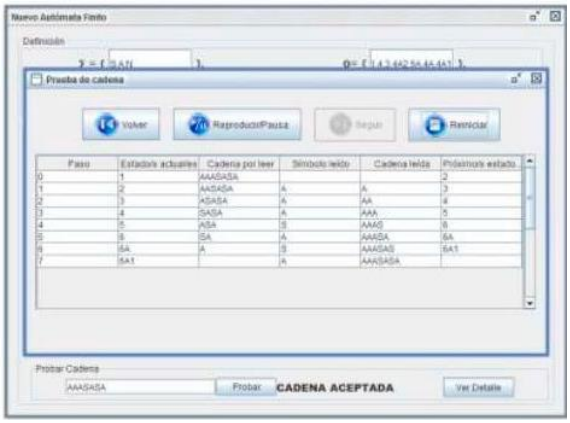

Figura 10. Pantalla informe resultado ejecución

De esta manera se logra control y trazabilidad de los cambios en los escenarios de los requerimientos funcionales.

3.3.6. Simulador y Validador de Redes de Petri Coloreadas.

La herramienta CPN Tool [20] es una herramienta gráfica que permite la simulación de Modelos de Redes de Petri Coloreadas (ver Figura 11).

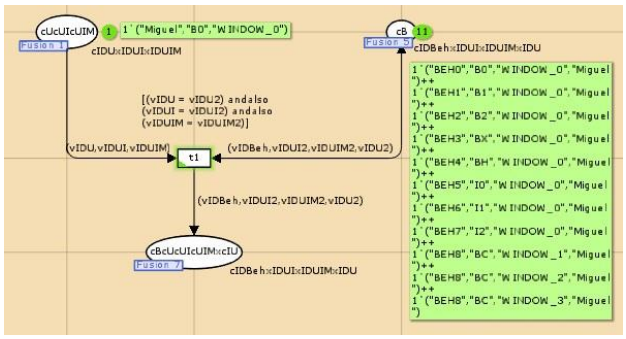

Figura 11. Un ejemplo de una red de Petri Coloreada (CPN).

Para comprender como se simula el funcionamiento de una interfaz, se desarrollará un ejemplo. En la Figura 11, se representa el funcionamiento de página web. El nodo superior izquierdo o lugar o place (1) representa a la interfaz con la que el usuario interactúa. Se puede observar un token a su derecha, representado por rectángulo, que contiene tres cadenas de texto. El mensaje en este caso indica que un usuario ha interactuado con la interface presionando un botón de una ventana (Window_0). El nodo superior derecho o lugar o place (11) representa el comportamiento de la interface, según el botón que se presione para cada usuario que hace uso de la misma. La transición denominada "t1" se encuentra habilitada, puesto que es posible la selección de un token de cualquiera de los dos lugares de entrada. Si se observan los tokens de los dos nodos, se puede ver que existen cadenas de ambos token que coinciden, independientemente del lugar que ocupen en la cadena (por ejemplo BO, WINDOW_0 y Miguel). Al ejecutarse "t1" se crea un token nuevo determinado por la inscripción del arco de salida. La interacción con la interfase definida en el comportamiento BEH0.

Una vez terminada la ejecución, el token con el comportamiento seleccionado se devuelve al nodo o lugar de origen. Cuando hay arcos de entrada y salida a un nodo o lugar con la misma inscripción, se representan como un arco bidireccional.

CPNTools es una herramienta que permite la definición, simulación y análisis de redes de Petri coloreadas. Permite realizar simulaciones automáticas, interactivas, o ambas. Las simulaciones automáticas son empleadas para analizar la red. Las simulaciones interactivas permiten seleccionar que transición se desea disparar, y el conjunto de tokens a contemplar para dicha simulación. Este último tipo de simulación se suele emplear para depuración de redes. Las simulaciones mixtas son para el monitoreo de la ejecución, a través de la implementación de agentes o monitores programados que graban el estado de la red, baja determinadas condiciones.

4. Conclusiones

A través de esta propuesta metodológica y sus herramientas de soporte, que se sintetizan como un conjunto de transformaciones aplicadas sobre el modelo conceptual primario, es posible generar nuevos modelos que sirvan para representar las máquinas abstractas necesarias para la validación de los requerimientos funcionales iniciales como así también de sus interfaces, garantizando de esta forma que los modelos reflejen fielmente la realidad, sin ambigüedades, manteniendo la coherencia y asegurando la trazabilidad a lo largo de todo el proceso de gestión de requerimientos.

Estas validaciones y simulaciones a las Máquinas Abstractas generadas a través de un proceso automatizado de transformaciones, ya sean sobre: Procesos de Negocios, o Plantillas de Casos de Uso, nos permiten confirmar las características deseables sobre las especificaciones de los requisitos funcionales del sistema a construir.

Además, la metodología planteada permite al especialista definir especificaciones de usabilidad en etapas tempranas del desarrollo de software en forma organizada, flexible, escalable y acorde a estándares de calidad vigentes. Esto queda demostrado a través del desarrollo metodológico propuesto como proceso, donde se emplean metodologías de Modelado de Negocios y de Casos de Uso, vinculadas a través del uso de conceptos y prácticas del paradigma del Desarrollo de Software Dirigido por Modelos.

De este modo, el proceso de validación propuesto resulta útil para validar el modelo conceptual y sus interfaces, para posteriormente construir el sistema de software que dará soporte al sistema de información.

5. Trabajos Futuros.

Como trabajos futuros, se seguirá investigando sobre modelos abstractos estandarizados para el desarrollo de interfaces, desde el punto de vista del Desarrollo de Software Dirigido por Modelos, teniendo en cuenta la incorporación de aspectos no funcionales en el modelado conceptual en forma temprana.

6. Referencias Bibliográficas.

[1] Pressman, R. S.. What a Tangled Web We Weave. IEEE Software, 17(1):18–21, Jan. 2000.

[2] Abrahao S., Condori-Fernandez N., Olsina L., and Pastor O., "Defining and validating metrics for navigational models," Australia, 2003.

[3] Norma ISO/IEC ISO9126-1, "Software Engineering - Product Quality - Part 1," 2001.

[4] Nigel Bevan, "Quality and usability: A new framework," Achieving software product quality, 1997.

[5] Mario G. Piattini, Felix O. Garcia, and Ismael Caballero, "Calidad de Sistemas Informáticos," México, ISBN 978-970-15-1267-8, 2007.

[6] ISO/IEC 25000, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE).

[7] L Bass and B John, "Linking usability to software arquitecture patterns through general scenarios," The journal of systems and software, no. 66, pp. 187 - 197, 2003.

[8] Eelke Folmer and Jan Bosh, "Architecting for usability: A survey. ," Journal of Systems and Software, pp. 61 - 78, 2004.

[9] Stephen J Mellor, Kendall Scott, Axel Uhl, and Dirk Weise, Model-Driven Architecture. Berlin / Heidelberg: Springer, 2002.

[10] MDA_Guide_Version1-0. 2003. [Online]. http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf

[11] G. Rossi and D. Schwabe, "Modeling and Implementing Web Applications using OOHDM," in Web Engineering, Modeling and Implementing Web Applications.: Springer, 2008, pp. 109-155.

[12] S. Daniel, P. Rita de Almeida, and M. Isbela, "OOHDM-Web: an environment for implementation of hypermendia applications in the WWW," in SIGWEB News l.8, 2, 1999, pp. 18-34.

[13] Nora Koch and Martin Wirsing, Software Engineering for Adaptative Hypermedia Applications. München, Germany: Ludwig-Maximilians University of Munich, 2000.

[14] N. Koch, A. Knapp, G. Zhang, and H. Baumeister, "UML-Based Web Engineering, An Approach Based On Standar.," in Web Engineering, Modelling and Implementing Web Applications..: Springer, 2008, pp. 157-191.

[15] Oscar Pastor and Juan Carlos Molina, Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling , Inc. Secaucus, NJ, Springer-Verlag New York, Ed. Valencia, USA, 2007.

[16] J. Gómez and C. Cachero, "OO-H Method: extending UML to model web interfaces.," in In information Modeling For internet Applications,. Hershey, PA.: Ed. IGI Publishing, , 2003, pp. 144-173.

[17] P.V., Albert M., and Pastor O. Fons J.,.: LNCS. Springer, 2003, vol. 2813, pp. 232-245.

[18] S. Ceri, P Fraternali, and A. Bongio, "Web Modeling Language (WebML): a modeling language for designing Web sites.," in 9th. World Wide Web Conference, 2000, pp. 137-157.

[19] Object Management Group. Business Process Modeling Notation (BPMN).

[20] Página web de CPNTools: http://wiki.daimi.au.dk/cpntools/cpntools.wiki