Metodología y herramienta de soporte para validar modelos conceptuales a través de máquinas abstractas

Marcelo Martín Marciszack[1], Manuel Perez Cota[2] y Mario Alberto Groppo[3]

Resumen

El presente trabajo, propone la definición de una metodología y herramienta de soporte asociada, para la especificación y validación de un Modelo Conceptual, a través de la transformación de modelos a autómatas finitos y su validación correspondiente. Describe las características del Proceso de Modelado, a través de la aplicación del Desarrollo dirigido por modelos MDD, con utilización de BPMN para el modelado de Procesos de Negocios. La gestión del proceso se realiza a través de una herramienta automatizada de Casos de Uso, su definición y posterior transformación a Máquinas de estado junto con la propuesta de controles a realizar sobre el modelo abstracto resultante, para de esta manera, validar el modelo de proceso de negocio que representa las especificaciones funcionales en el modelo conceptual del dominio bajo estudio.

Palabras clave: modelado conceptual, especificaciones funcionales, validación de modelos, autómatas finitos, desarrollo de software dirigido por modelos.

1 Facultad Regional Córdoba, Universidad Tecnológica Nacional, Argentina.
2 Universidad de Vigo, España.
3 Argentina

Abstract

This paper proposes the definition of a methodology and associated support tool for the specification and validation of a Conceptual Model through the transformation of models to finite automata and the corresponding validation. It provides a description of the characteristics of the Modeling Process through the application of model-driven development MDD, with use of BPMN for modeling Business Processes. Process management is performed through an automated Use Case tool, its definition and subsequent transformation to state machines along with the definition of controls to be performed on the resulting abstract model, in order to validate the business process model which represents the functional specifications in the conceptual model of the domain under study.

Keywords: conceptual modeling, functional specification, validation of models, finite automata, model driven software development.

1. Introducción

Las debilidades de la mayoría de los métodos y metodologías utilizados para la obtención de esquemas conceptuales se reflejan en las primeras etapas del proceso de desarrollo de software. El principal problema derivado de estas debilidades metodológicas radica en la dificultad en determinar si el modelo conceptual refleja fiel y completamente la esencia del dominio (Insfrán, Díaz, & Burbano, 2002).

La preocupación por definir los requisitos de manera adecuada está extensamente tratada en Boehm & Basili (2001), donde el eje central es la definición de buenas prácticas en el establecimiento de las mismas, ya que plantea que el éxito de cualquier proyecto de desarrollo está íntimamente relacionado con la calidad de los requisitos.

Es así que la presente propuesta abarca la definición de una metodología, junto con un conjunto de herramientas de soporte de procesos, y la definición de transformaciones automatizadas entre los modelos intermedios, que posibilita validar y verificar si el modelo conceptual construido representa fielmente el sistema de información a construir (Sommerville, 2011; Sesé Muniátegui, 2007).

2. Modelado conceptual y requerimientos funcionales

Desde la óptica disciplinar de los Sistemas de Información y los sistemas de software asociados a estos, un Esquema Conceptual será definido como un modelo de representación de la realidad, sobre un dominio de problema determinado, el cual deberá incluir además, el lenguaje utilizado en su definición, de manera que no existan ambigüedades, de esta manera de reducir el "gap" semántico, entre el constructor del modelo y los usuarios del mismo.

En este contexto el presente trabajo, se focaliza con la visión aportada por Insfrán, Tejadillos, Marti, & Burbano (2007) en donde un Esquema Conceptual es interpretado como un refinamiento de los requerimientos de usuario a través de los requisitos funcionales que resultarán en especificaciones más detalladas que constituirán dicho esquema.

El Modelo Conceptual estableciendo los requisitos funcionales de un sistema de información (Letelier, Sanchez, & Ramos, 1999), es la pieza clave para establecer el vínculo entre el espacio del problema y el espacio de la solución. Las deficiencias del modelo conceptual tienen un impacto considerable en las posteriores actividades en el proceso de desarrollo de software.

3. Tendencias actuales en la construcción de modelos

En los últimos años, el modelado de procesos de negocios, ha despertado especial interés por parte de la Ingeniería de Software, debido a que brinda un punto de partida para la captura de requisitos. Estos modelos se consideran esenciales para conocer las actividades de una organización, permitiendo establecer los fundamentos para la construcción de un sistema de información correcto.

La OMG ha utilizado para representar los modelos de negocio diferentes tipos de notaciones, pero le ha dado principal importancia a Business Process Modeling Notation (BPMN) (2009), y a Unified Modeling Language (UML)(2002) (a través de los diagramas de actividad y diagrama de casos de uso). Ambas notaciones ofrecen soluciones similares para la mayoría de los patrones de flujo de trabajo que soportan. Esto es lógico debido a que ambos estándares fueron diseñadas para satisfacer las mismas necesidades de modelado, pero con objetivos diferentes en diferentes etapas del desarrollo.

El interés de la Ingeniería de Software en el Modelado del Negocio surge porque a partir del estudio de la transformación de modelos, es posible iniciar el modelado de sistemas de información (elicitación de requisitos) que se pueden integrar al proceso de desarrollo del software.

Para el desarrollo de esta propuesta nos centraremos en el Desarrollo de Software Dirigido por Modelos, (MDD) (Pons, Giandini, & Pérez, 2010), las definiciones y documentos emitidos por la OMG, (XML Metadata Interchange (XMI), 2007) , (MOF Query / Views / Transformations, 2008) que es el organismo que se ha encargado del estudio y definición de los procesos de transformación de los modelos, en forma conjunta con el World Wide Web Consortium (Extensible Markup Language (XML), 2008).

4. Propuesta metodológica

Destacando el rol fundamental que actualmente desempeñan los modelos en el proceso de desarrollo del software, la propuesta metodológica inicia con el modelado del negocio a través de la notación BPMN. Una vez que se obtiene esta representación, se continúa identificando aquellas actividades que implican un manejo inherente de información para posteriormente transformarlas en casos de uso. Seguidamente se debe convertir cada caso de uso en una máquina abstracta, proceso que es soportado por la herramienta SIAR (Sistema Integral para la Administración de Requerimientos). Finalmente se procede a verificar la consistencia secuencial de cada escenario del caso de uso para de este modo detectar cualquier tipo de anomalía.

A. Justificación de la propuesta

El proceso de creación y mantenimiento de Modelos Conceptuales es una actividad que se realiza generalmente en forma manual, generando con gran frecuencia, inconsistencias entre modelos.

Estas inconsistencias impactan de forma negativa en la trazabilidad de los requerimientos, y adicionalmente, dificultan su análisis para verificar la validez de los mismos.

La idea central consiste en observar el sistema de software a construir como un producto completo y a su proceso de construcción como un trabajo ingenieril (Pons, Giandini, & Pérez, 2010). Es decir, un proceso planificado basado en metodologías formales apoyadas por el uso de herramientas.

B. Fundamentación.

La abstracción es una de las principales técnicas con la que la mente humana se enfrenta a la complejidad. Ocultando lo que es irrelevante, un sistema complejo se puede reducir a algo comprensible y manejable.

El modelado del negocio es un punto fundamental para comprender el contexto del sistema que se está construyendo, y esto impacta directamente en el éxito o fracaso de un proyecto de software. Si no podemos entender el negocio, se pueden presumir conceptos erróneos sobre lo que debe hacer el software y cómo debe ser utilizado.

C. Etapas 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.

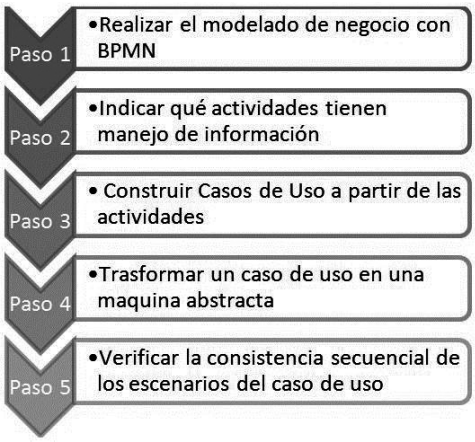

Figura 1. Etapas de aplicación de la Metodologías

1) Realizar el Modelado de Negocio con BPMN

Business Process Modeling Notation o BPMN, en español Notación para el Modelado de Procesos de Negocio, es una notación gráfica estandarizada que permite el modelado de procesos de negocio en un formato de flujo de trabajo (workflow).

Modelar un Proceso de Negocio implica representar cómo una empresa realiza sus objetivos centrales; los objetivos por si mismos son importantes, pero por el momento no son soportados por la notación. Con BPMN, solo los procesos son modelados.

En esta etapa, siguiendo la metodología de modelado de BPMN, se debe realizar el modelado del negocio identificando los Procesos de Negocio involucrados y la secuencia de actividades que se ejecutan en cada proceso.

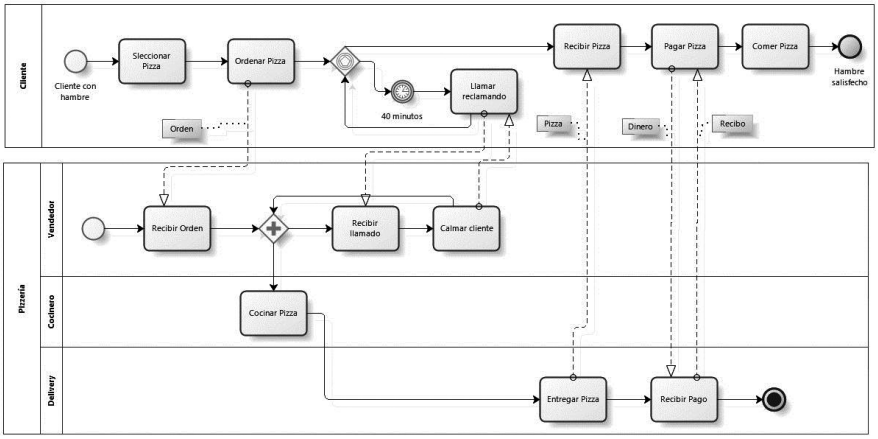

Figura 2. Ejemplo de un Modelo de Procesos de Negocio modelado con BPMN.

2) Indicar qué actividades tienen manejo de información.

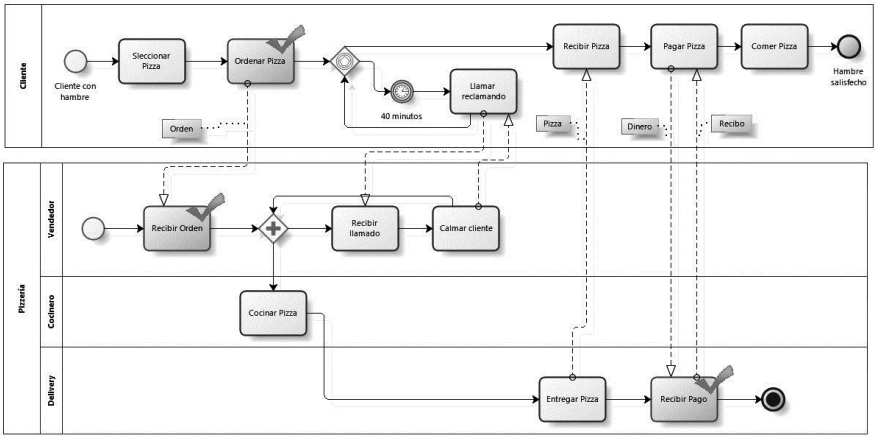

Figura 3. Se seleccionan las actividades automatizadas que formarán parte del sistema de Información quedando el resto como manuales.

Aquí, el analista deberá identificar en los diagramas de procesos aquellas actividades que utilicen / generen información, diferenciándolas de aquellas que son puramente manuales.

El conocimiento y la capacidad del analista serán de gran importancia en esta etapa.

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, 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.

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.

4) Trasformar un caso de uso en una maquina abstracta (AF).

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

Entonces, una vez que se cuenta con una especificación detallada de los Casos de Uso que satisfacen las necesidades informáticas del negocio, en el paso 4 se debe realizar la trasformación de los mismos a maquinas abstractas.

Para formalizar esta transformación se definieron un conjunto de reglas de conversión del Caso de Uso en un grafo de estados:

- Todo caso de uso tiene al menos un escenario principal.
- Un caso de uso puede no tener ningún, o tener uno o tener varios escenarios alternativos.
- Cada paso de un escenario de un caso de uso responde a un estado y es un nodo de la máquina de estados.
- Todo caso de uso tiene un paso inicial, y es el primer paso de todos los escenarios. Este paso es el estado/nodo inicial.

- Todo caso de uso tiene un paso final por aceptación y es el último paso del escenario principal. También puede ser el último paso de algún escenario alternativo. Este paso es el estado/nodo final por aceptación.
- Un caso de uso puede no tener ningún final por error, o tener uno o varios finales por error, en cuyo caso son el último paso de un escenario alternativo. Estos pasos se denominan estados/nodos finales por error.
- Todo caso de uso pasa de un estado/nodo a otro por medio de una función de transición, que es una tabla de transición de estados donde se muestra a qué estado se moverá un autómata finito dado, basándose en el estado actual y otras entradas.
- Partiendo de un estado/nodo origen se puede continuar en un único estado/nodo destino, o en dos nodos/estados destino alternativos dependiendo de una condición de bifurcación.

5) Verificar la consistencia secuencial de los escenarios del caso de uso.

Al contar con la representación de cada Caso de Uso como si fuese un Autómata Finito (AF), y tal lo desarrollado por Pérez Cota, Groppo, & Marciszack (2013), cada uno de éstos se obtiene a partir de una transformación directa de la representación del modelo de Proceso de Negocio 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 y debe 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 re invocación 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.

5. Herramientas de soporte

Tiene por objetivo la captura de requerimientos funcionales, es una herramienta web, tiene por finalidad dar soporte a la construcción del modelo de requerimientos de un proyecto de software según el estándar UM, en la definición y construcción de los casos de uso.

A. SIAR (Sistema Integral de Administración de Requerimientos)

Los casos de usos son una herramienta de generación y análisis de requisitos de sistemas. La finalidad principal de SIAR es la administración de casos de uso con una herramienta informática que agilice su registración, normalice su contenido y posibilite implementar validaciones funcionales, como por ejemplo un método automatizado de análisis de consistencia de Casos de Uso, para lo cual el sistema genera un grafo con la transición de estados de cada Caso de Uso, expresado en el protocolo XPDL (Lenguaje de Definición de Flujo de Trabajo), que es analizado en un simulador de autómata finito determinista para verificar la cohesión de los escenarios en él definidos.

B. Etapas de la Metodología a las que da soporte

En la actualidad SIAR Vers. 1.1. es capaz de dar soporte a las etapas 3 y 4 de la metodología propuesta en forma directa, como se detalla a continuación, y deja el Autómata Finito Generado en el formato, que la herramienta de validación de Máquinas abstractas necesita para validar el Autómata y por ende el caso de Uso que representa, que es la etapa 5 de la Metodología propuesta:

1 Etapa 3 – Construir Casos de Uso a partir de las actividades

SIAR es una aplicación web que permite registrar en forma normalizada los casos de uso y cuya primera versión comprende el siguiente alcance:

- 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, pre-condiciones, post-condiciones, 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.
- Diseño y validación del Modelo Conceptual.
- Gestión de atributos de procesos de negocio, de actividades de negocio que los componen y los casos de uso asociados a estas actividades.
- Registración y consulta de un glosario por proyecto, con entradas y sinónimos, siguiendo las recomendaciones de LEL (Léxico Extendido del Lenguaje), que es una estrategia de modelado de requisitos basada en lenguaje natural.

Figura 4. Versionado de Proyectos y Casos de Uso en SIAR

2) Etapa 4 - Trasformar un caso de uso en una maquina 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, SIAR 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:

- A = 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.

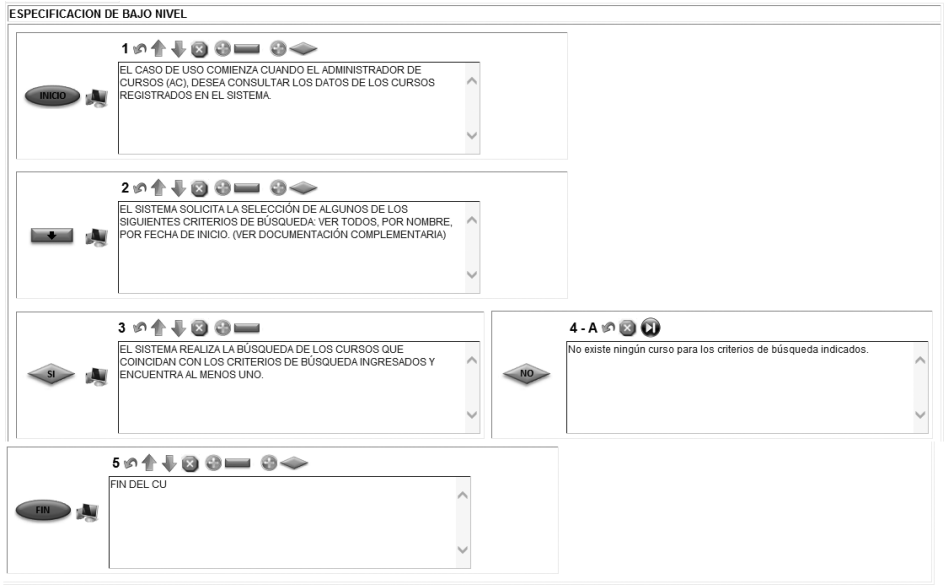

Figura 5. Bifurcación en la especificación de bajo nivel de un caso de uso.

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.

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 6. Tabla de estados de un caso de uso. Transformación automática por SIAR.

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.

```
<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 7. Fragmento de archivo XPDL generado por SIAR.

C. Simulador de Autómata Finito

Esta herramienta de validación de Autómatas Finitos, por el momento no está integrado funcionalmente al SIAR. Gracias a este simulador es posible llevar a cabo la última etapa del proceso metodológico que es Verificar la consistencia secuencial de los escenarios del caso de uso.

Lo que se hace es ingresar el archivo XPDL (salida de SIAR), que representa al caso de uso como grafo de estados, al programa "Autómata Finito".

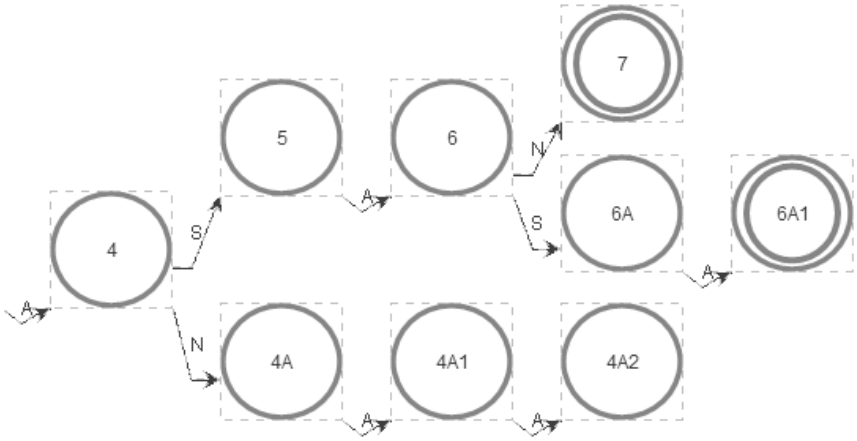

Figura 8. Grafo de estados en el simulador de autómatas finitos

D. 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 AF:*** 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 Paralelo) 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.

El simulador posibilita probar el grafo de estados y comprobar si es aceptado o rechazado. Si el AF produce un rechazo, indica que en algunos de los caminos hay un error en la secuencia y se puede visualizar el detalle para detectar la inconsistencia, la cual puede ser corregida en una nueva versión del caso de uso, generando un nuevo grafo de estados automáticamente.

6. Conclusión

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, 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.

Por tal motivo, este proceso de validación propuesto resulta sumamente útil para validar el modelo conceptual propuesto para luego construir el sistema de software que de soporte al sistema de información.

Bibliografía

Boehm, B., & Basili, V. (2001). Software defect reduction top 10 list. *IEEE Computer*.

Business Process Modeling Notation (BPMN). (2009, enero 3). Retrieved from Object Management Group: http://www.omg.org/technology/documents/br_pm_spec_catalog.htm

DeMarco, T. (1978). *Structured Analysis and System Specification.* Yourdon Press.

Extensible Markup Language (XML). (2008, noviembre 26). Retrieved from World Wide Web Consortium: http://www.w3.org/XML/

Insfrán, E., Díaz, I., & Burbano, M. (2002). *Modelado de Requisitos para la Obtención de esquemas conceptuales*. Retrieved from http://www.dsic.upv.es/~einsfran/papers/39-ideas2002.pdf

Insfrán, E., Tejadillos, E., Marti, S., & Burbano, M. (2007, diciembre 12). *Transformación de Especificación de requisitos en esquemas conceptuales usando Diagramas de Interacción.* Retrieved from inf.puc-rio.br: www.inf.puc-rio.br/~wer02/zip/Transformacion_Espec(7).pdf

Letelier, P., Sanchez, P., & Ramos, I. (1999). *Un ambiente para especificaciones incremental y validacin de modelos conceptuales.* Retrieved marzo 12, 2013, from researchgate.net: http://www.researchgate.net/publication/36720988_Un_ambiente_para_especificaciones_incremental_y_validacin_de_modelos_conceptuales/file/d912f50ca20c33f5e5.pdf

MOF Query / Views / Transformations. (2008, abril). Retrieved from Object Management Group: http://www.omg.org/technology/documents/modeling_spec_catalog.htm

Pérez Cota, M., Groppo, M. A., & Marciszack, M. (2013). Validación de Especificaciones Funcionales en el modelado de Esquemas Conceptuales a través de Máquinas Abstractas. *CoNaIISI.* Córdoba.

Pons, C., Giandini, R., & Pérez, G. (2010). *Desarrollo de Software dirigido por modelos – Conceptos Teóricos y su aplicación práctica.* Universidad Nacional de la Plata.

Sesé Muniátegui, F. (2007, febrero 16). *Tesis Doctoral: Propuesta de un método de validación de esquemas conceptuales y análisis comparativo de la noción de información en los métodos de desarrollo de Sistemas de información.* Retrieved mayo 20, 2008, from tesisenxarxa.net: www.tesisenxarxa.net/TDX-0517107-131929/

Sommerville, I. (2011). *Ingeniería de Software.* México: Pearson Educación.

Unified Modelling Language: Superstructure. (2002, julio). Retrieved from Object Management Group: http://www.omg.org

XML Metadata Interchange (XMI). (2007, diciembre 1). Retrieved from Object Management Group: http://www.omg.org/technology/documents/modeling_spec_catalog.htm#XMI