risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido / Recibido: 04/07/2012
Aceitação / Aceptación: 04/12/2012

Modelamiento semántico con Dinámica de Sistemas en el proceso de desarrollo de software

Ricardo Vicente Jaime Vivas1

ricardojaime@udi.edu.co

[1] Grupo de Investigación en Tecnologías de la Información aplicadas a la Educación, Universitaria de Investigación y Desarrollo UDI. Calle 9 # 23-55, Bucaramanga (Colombia)

DOI: 10.4304/risti.10.19-34

Resumen: Este artículo propone el uso de la Dinámica de Sistemas para el modelamiento semántico en proyectos de desarrollo de software. El carácter interdisciplinario y el rigor matemático de la Dinámica de Sistemas, facilitan la integración entre stakeholders y desarrolladores en la construcción y validación de representaciones de conocimiento en el dominio del problema, que pueden ser transformadas en diagramas de diseño de software.

Palabras-clave: Dinámica de Sistemas, Sistemas de Información, Ingeniería de Requerimientos, Modelamiento Semántico, Objetos de Frontera.

Abstract: This paper proposes using System Dynamics for semantic modeling in software development process. Interdisciplinary nature and mathematical rigour of System Dynamics, facilitate integrating stakeholders and developers to build and validate knowledge representations about problem domain that can become software design diagrams.

Key-words: System Dynamics, Information Systems, Requirements Engineering, Semantic Modeling, Boundary Objects.

1. Introducción

Pese a los avances en Ingeniería del Software, se reportan cientos de millones de dólares en pérdidas en proyectos de software (Charette, 2005). La tasa de fracasos está entre el 66% (Cox, Niazi, & Verner, 2009) y el 90% compuesto por 50% de fallas totales y 40% de parciales (Tang & Huang, 2011). Aunque entre 1994 y 2008 la tasa de éxito en proyectos de software ha pasado del 16% al 32%, la de fracaso ha bajado de 34% a 21%, y la de rescate de proyectos con altos sobrecostos o con disminución de expectativas ha bajado del 53% al 44%, estas cifras todavía indican un 65% de proyectos con fallas parciales o totales (Nasir & Sahibuddin, 2011).

Para el 60% de las publicaciones técnicas entre 1991 y 2006, la determinación de requerimientos era el factor más determinante del éxito de un proyecto de software (Nasir & Sahibuddin, 2011). Se estima que el 56% de las fallas y el 82% de los costos en el desarrollo de software se deben a fallas en la definición de requerimientos (Williams, 2001), por incapacidad para manejar la complejidad de los proyectos, comunicación deficiente (Charette, 2005) y documentación inapropiada entre stakeholders y desarrolladores (Costain & McKenna, 2012). La sociología de la tecnología describe el proceso: 1) una demostración de la tecnología capta el interés de los stakeholders; 2) una ola de entusiasmo con requerimientos sobredimensionados; 3) una serie de fracasos que conlleva la cancelación de los proyectos; 4) rescate de una fracción de los proyectos tras una reducción sustancial de los requerimientos (Geels, 2007).

A partir de la problemática descrita, este artículo aborda la pregunta: ¿se puede utilizar el modelamiento con Dinámica de Sistemas para mejorar la determinación de requerimientos y el diseño en el proceso de desarrollo de software?

2. Problemática del proceso de desarrollo de software

En los modelos de desarrollo de software de los años 70 y 80, la determinación de requerimientos era una actividad de la fase de análisis, ejecutada en su totalidad antes de la fase de diseño, para que los requerimientos no fueran limitados por consideraciones de diseño o de disponibilidad de tecnología (Sommerville, 2005). En la década de los 90 la determinación de requerimientos deja de ser una actividad; en 1993 IEEE efectuó el primer Simposio Internacional y en 1994 el primer Congreso Internacional en Ingeniería de Requerimientos (IR).

La Ingeniería de Requerimientos (IR) se define como: proceso de especificación y refinamiento de requerimientos a partir de las necesidades de los stakeholders (Hofmann & Lehner, 2001); proceso de descubrimiento, documentación y mantenimiento de requerimientos durante todo el ciclo de vida de desarrollo de software (Napier, Mathiassen, & Johnson, 2005); descripción del dominio de un problema y de los efectos que el cliente desea ejercer sobre el mismo, y especificación de la forma en que las tecnologías de la información intervienen en el proceso (Cox, Niazi, & Verner, 2009). Comprende cinco etapas: 1) elicitación, identificación de fuentes de información; 2) análisis, comprensión de los requerimientos; 3) validación, revisión de los requerimientos con los stakeholders; 4) negociación, conciliación de los desacuerdos de revisión; y 5) documentación comprensible para stakeholders y desarrolladores (Sommerville, 2005).

El proceso tiene riesgos: 1) omisión de requerimientos esenciales; 2) imposición de intereses de los desarrolladores a los stakeholders; 3) omisión de requerimientos no funcionales tales como la validación matemática; 4) utilización de lenguaje técnico exclusivo de los desarrolladores; y 5) cierre definitivo de los requerimientos antes de iniciar la construcción del software (Lawrence, Wiegers, & Ebert, 2001). Los riesgos se originan en varios supuestos: 1) los stakeholders conocen bien sus necesidades y las comunican eficazmente; 2) dichas necesidades son estáticas y su determinación factible al comienzo del proyecto; y 3) la comunicación entre stakeholders y desarrolladores es fluida. Estos supuestos son errados (Ramiller & Wagner, 2012); con frecuencia los stakeholders no tienen claro lo que quieren ni las posibilidades que brinda la

tecnología, siendo trabajo de IR motivarlos a pensar más allá de las fronteras iniciales del problema (Robertson, 2005).

El Ingeniero de Requerimientos no es solo un relator de ideas, sino que coordina diversas áreas del conocimiento que convergen en el dominio de un problema, por lo que su ámbito no es la Ingeniería del Software sino la Ingeniería de Sistemas (Gonzales, 2005). Su visión enfatiza que los requerimientos no son acerca del software sino acerca del dominio del problema, siempre potencialmente complejo, que conlleva que el software generalmente es más complejo de lo que se prevé (Jackson, 2004). Desarrollar software no es escribir y documentar líneas de programación, sino comprender el dominio de un problema, un proceso de aprendizaje, comunicación y negociación, durante el cual el desarrollador debe poder integrar varios dominios de conocimiento antes de emprender la construcción del software (Curtis, Krasner, & Iscoe, 1988).

Por su contexto interdisciplinario, la IR debe estar mediada por el lenguaje natural o representaciones cercanas a este, y no por representaciones técnicas, como UML, comprensibles para los desarrolladores pero confusas para los stakeholders (Robertson, 2005) y que los llevan a perder interacción y comprensión suficiente del dominio del problema (Luna-Reyes, Black, Cresswell, & Pardo, 2008). Aunque el uso del lenguaje natural implica riesgos léxicos, sintácticos y pragmáticos que derivan en ambigüedad, algunos autores sugieren usar un lenguaje natural restringido que pueda contribuir a la automatización parcial de las tareas de análisis y verificación de requerimientos (Sawyer, Rayson, & Cosh, 2005).

Para sustentar la propuesta de este artículo, la IR se define como un proceso de aprendizaje en ambiente interdisciplinario, que genera representaciones en el dominio de un problema, con las cuales se formulan las características y el diseño del software con que se quiere actuar sobre dicho problema.

3. Dinámica de Sistemas

La Dinámica de Sistemas (DS) consiste en la formulación, prueba y depuración de explicaciones de las causas internas del comportamiento de un sistema, para el desarrollo de políticas de manejo y toma de decisiones, haciendo uso de mapas informales, modelos formales y simulaciones computarizadas que representan el origen endógeno del comportamiento del sistema (Richardson, 2011).

La Figura 1 muestra los modelos aludidos en dicha definición: a) el diagrama de influencias es el mapa informal en forma de grafo; los modelos formales son b) el diagrama de flujos y niveles y c) el modelo matemático; d) la simulación del comportamiento del sistema.

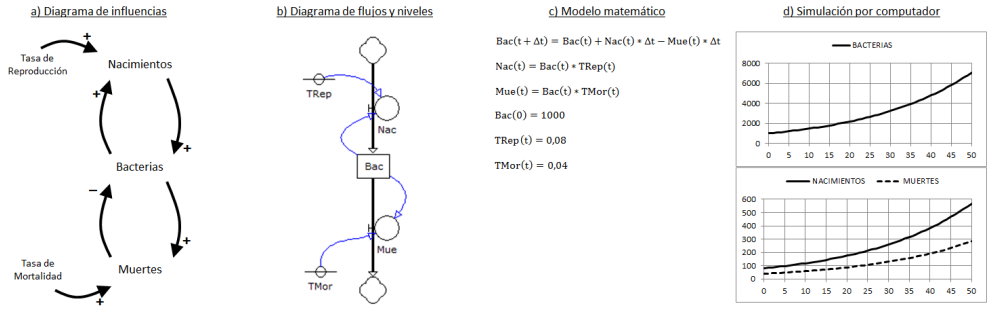

Figura 1 – Modelos usuales en Dinámica de Sistemas

El modelamiento con DS es el desarrollo iterativo de estos cuatro modelos, hasta que se obtiene una representación del sistema cuya simulación por computador reproduce el comportamiento observado en la realidad, de manera que permite probar políticas de manejo del sistema. El proceso consta de tres fases: 1) la conceptualización, que consiste en la descripción verbal del sistema y la construcción de un diagrama de influencias; 2) la representación o formulación, que es la construcción del diagrama de flujos y niveles y el modelo matemático de simulación; y 3) el análisis y evaluación del modelo, consistente en la revisión de los resultados de la simulación, la formulación y prueba de políticas de manejo del sistema (Aracil & Gordillo, Dinámica de Sistemas, 1997).

3.1. Modelamiento cuantitativo y cualitativo.

La tendencia ortodoxa suele evitar los diagramas de influencias. Jay Forrester, fundador de la DS, afirma que carecen de una disciplina de pensamiento, fallan en la identificación de los elementos que producen el comportamiento del sistema, y llevan a apreciaciones erróneas, por lo que solo sirven como elementos explicativos de modelos previamente representados en flujos y niveles (Forrester, 1994). Pero para la tendencia cualitativa estos diagramas pueden ser desarrollados con el suficiente rigor, y ayudan a las personas a externalizar y enriquecer sus modelos mentales (Wolstenholme, 1999). Son útiles para resumir grandes cantidades de texto, orientar una discusión, explicar el comportamiento de los sistemas a partir de estructuras cíclicas, ampliar el contexto de abordaje de un problema, y como paso intermedio entre el lenguaje natural y el lenguaje matemático (Coyle, 2000). Apoyan el trabajo interdisciplinario del proceso: los diagramas de influencias son construidos mientras que personas de diferentes disciplinas discuten acerca de un problema que los involucra, y son representaciones cercanas al lenguaje natural, en las que diferentes perspectivas expresan sus argumentos.

La DS es desde lo cuantitativo una plataforma para la estrategia, y desde lo cualitativo una forma de pensamiento para la representación y el aprendizaje (Andrade, Dyner, Espinosa, López, & Sotaquirá, 2001).

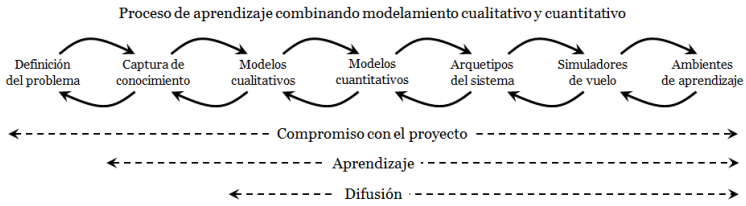

Figura 2 – Combinación de modelamiento cualitativo y cuantitativo. Adaptado de (Wolstenholme, 1999, pág. 425)

El modelamiento cualitativo y el cuantitativo, integrados en un proceso de aprendizaje sobre un problema, se pueden constituir en modelamiento semántico.

3.2. Modelamiento semántico.

El modelamiento semántico es la actividad de representación del significado (Date, 1998). Date lo aborda desde el diseño de bases de datos: los modelos conceptuales, tales como el modelo relacional, aportan una comprensión limitada del significado de los datos y de algunas de sus interrelaciones.

En el contexto del desarrollo de software y la problemática alrededor de los requerimientos y el diseño, se puede apreciar la importancia del modelamiento semántico: salvo que tengan una preparación específica en desarrollo de software, difícilmente los stakeholders podrán expresar opiniones críticas con respecto a representaciones de diseño como diagramas relacionales o diagramas de clases. Modelamiento semántico en un proyecto de software es la representación del conocimiento en el dominio del problema, en un lenguaje más cercano a los stakeholders que de los desarrolladores. Apunta justamente a una de las recomendaciones mencionadas en la problemática del desarrollo de software.

3.3. Perspectivas de Dinámica de Sistemas en modelamiento semántico

Abdel-Hamid simula los efectos de diferentes políticas de aseguramiento de calidad en el costo de desarrollo de software (Abdel-Hamid, 1988). Madachy simula de forma similar estrategias que combinan aspectos técnicos, económicos y organizacionales (Madachy, 2008). Desde la tendencia cuantitativa de DS se han hecho múltiples modelos orientados a la optimización de recursos para el proceso de desarrollo de software (Georgantzas & Katsamakas, 2008). En otros casos se ha propuesto aprovechar las características del modelamiento cuantitativo con DS en escenarios de aprendizaje colaborativo, en el que diferentes especialistas comparten y discuten su conocimiento, a medida que negocian el significado de los datos que debe manejar el software (Luna-Reyes, Black, Cresswell, & Pardo, 2008).

La construcción de significados traza un vínculo con el modelamiento semántico, por la utilización que se puede hacer de los diagramas de DS como objetos de frontera semánticos, en el sentido de proveer a las personas un método para expresar su conocimiento en el dominio de un problema, a través de un modelo o mapa estandarizado (Carlile, 2002) y establecer acuerdos entre diversos campos del conocimiento acerca de los significados comunes creados en el modelamiento (Carlile, 2004). Un modelo con DS, cumpliendo algunas restricciones de estandarización

durante su construcción, podría constituirse en objeto de frontera semántico, es decir, un acuerdo interdisciplinario sobre la mejor representación posible del dominio de un problema, una traducción a un lenguaje común que puede ayudar a la construcción colectiva de los requerimientos a partir de la comprensión del dominio del problema (Loucopoulos & Preskas, 2003).

El alto nivel de transdisciplinariedad, generalidad y escalabilidad de la DS, que la hace aplicable casi en cualquier área del conocimiento, y su orientación a la diagramación, promueven el aprendizaje colaborativo (Schwaninger & Ríos, 2008), y permiten afirmar que un diseño computacional derivado de modelos con DS, no es la interpretación particular del diseñador, sino la transformación de un acuerdo suyo con los stakeholders, en requerimientos y diseños para el software.

El modelamiento semántico puede generar un impacto favorable tanto para la DS como para la Ingeniería de Sistemas. Si se consigue transformar modelos realizados con DS en objetos de frontera, diseños computacionales, bases de datos y líneas de programación, se superaría el factor adverso de que el modelamiento con DS usualmente se queda en el terreno de la comprensión (Roberts, 2007), interesante solo para los stakeholders pero no para los desarrolladores, centrado en la toma de decisiones mas no en su implementación. La DS puede constituir una forma de intervención organizacional mediada por la tecnología informática (Größler, 2007).

4. Modelamiento semántico con Dinámica de Sistemas

4.1. Antecedentes.

Para la construcción de un software de gestión de documentos, Tignor toma un diagrama de actividades y le aplica transformaciones hasta convertirlo en diagrama de flujos y niveles, a partir del cual formula un modelo matemático que utiliza para simular el proceso y probar políticas de gestión documental, que luego incorpora al software (Tignor, 2003). Como conclusión afirma que ninguno de los modelos de UML representa ni la estructura ni el comportamiento de un sistema, por lo que la DS es más conveniente para el modelamiento del problema. Señala que una desventaja de UML frente a DS es que no permite la simulación del sistema (Tignor, 2004). Esta propuesta sin embargo no usa la DS estrictamente para modelamiento semántico, sino para validar un diseño previo en UML.

Chang y Tu sugieren convertir modelos en DS en diseños orientados a objetos en UML, estableciendo equivalencias desde el diagrama de flujos y niveles: sector–clase; nivel–atributo; flujo–método y variable auxiliar–atributo (Chang & Tu, 2005). Su modelamiento matemático es de tendencia econométrica; un modelamiento estructurado generaría más métodos y menos atributos. No se encontraron desarrollos posteriores de los autores sobre el tema.

Hurtado pretende transformar el diagrama de flujos y niveles en diagramas de clases en arquitectura cliente servidor (Hurtado, 2010). Pero su propuesta es solo la implementación de un modelo de simulación en arquitectura cliente servidor; todos los elementos, tanto flujos como niveles se convierten en clases; y en los ejemplos que

desarrolla solo formula la ecuación de los niveles, mientras que en las demás abusa de la generación de valores por método aleatorio.

4.2. Metodología propuesta.

Esta propuesta es pertinente en problemas en cuyo abordaje convergen expertos en diversas disciplinas, no directamente relacionadas con los saberes de los desarrolladores de software, y cuya gestión implique toma de decisiones con fundamento en cálculos. Consta de los siguientes pasos: 1) formulación del problema; 2) modelamiento cualitativo de la estructura básica del sistema; 3) modelamiento cuantitativo; 4) modelamiento enriquecido del sistema; 5) transformación de modelos cualitativos y cuantitativos. A continuación se explican estos pasos mediante un ejemplo, de alcance limitado para mantener su sencillez.

Formulación del problema.

El ejemplo consiste en el desarrollo de un sistema de información para una granja dedicada a la compra de animales, su manutención y venta. El único cuidado que se considera es proveerles alimento a los animales. Los egresos son causados por las compras de animales y alimento, y los ingresos se obtienen de la venta de animales. Por ser un modelo didáctico, no se incluyen elementos como las enfermedades de los animales, otros cuidados durante su ciclo en la granja, pérdidas de alimento por deterioro, ni otros egresos como los asociados a la gestión administrativa.

Modelamiento cualitativo de la estructura básica del sistema.

Este paso consiste en la elaboración de un diagrama de influencias restringido a dos tipos de elementos: sustantivos y verbos. Los sustantivos son magnitudes medibles en el sistema: ¿cuántos animales hay?, ¿cuánto alimento hay?, ¿de cuánto capital se dispone? ¿cuáles son sus unidades de medida?. Los verbos son acciones o eventos que pueden cambiar de valor de los sustantivos, y que se miden en las mismas unidades de aquellos pero con relación a la unidad de tiempo. En este paso es importante la capacidad del modelador para identificar elementos durante el abordaje del problema con los stakeholders.

Cada sustantivo conforma un subsistema (algunos autores lo denominan sector) con los verbos de los que recibe influencia. La Figura 3 muestra los tres subsistemas implicados: 1) animales, compra de animales y venta de animales; 2) alimento, compra de alimento y consumo de alimento; y 3) capital, egresos e ingresos. Es importante mencionar la importancia del proceso gradual de construcción del diagrama, y la argumentación para la inclusión de cada elemento.

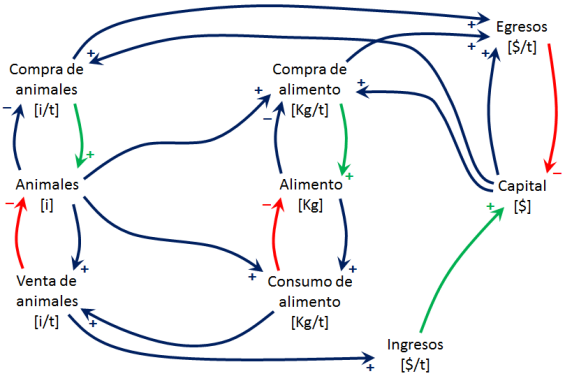

Figura 3 – Modelamiento cualitativo de la estructura básica del sistema

En DS la complejidad está dada por el número de subsistemas, las relaciones entre ellos, y los ciclos o secuencias cerradas de relaciones que integran varios subsistemas. Entre otros la Figura 3 contiene el ciclo Capital - Compra de animales - Animales - Venta de animales - Ingresos - Capital.

Tabla 1 – Clasificación de elementos de la estructura básica del sistema

Nombre	**Abreviatura**	**Tipo**	**Unidad**	
Animales	ANI	Sustantivo	Individuo	[i]
Alimento	ALI	Sustantivo	Kilogramo	[Kg]
Capital	CAP	Sustantivo	Monetaria	[$]
Compra de animales	CAN	Verbo		[i/t]
Venta de animales	VAN	Verbo		[i/t]
Compra de alimento	CAL	Verbo		[Kg/t]
Consumo de alimento	CON	Verbo		[Kg/t]
Egresos	EGR	Verbo		[$/t]
Ingresos	ING	Verbo		[$/t]

Las relaciones que parten de un verbo hacia un sustantivo llevan signo "+" cuando incrementan el valor del sustantivo y "-" cuando lo disminuyen. Las que parten de un sustantivo y llegan a un verbo dentro del mismo subsistema, implican límites, posibilidades o causalidades, directa o inversamente proporcionales según su signo sea "+" o "-" respectivamente. No se permiten relaciones entre sustantivos; un sustantivo solo puede influir sobre otro a través de los verbos.

Tabla 2 – Relaciones entre elementos del sistema

De – a	Signo	Descripción
De sustantivo a verbo en el mismo subsistema		
Animales - Compra de animales	-	Si hay muchos animales la compra debe disminuir; si hay pocos animales la compra debe aumentar.
Animales - Venta de animales	+	Si hay animales, se pueden vender; si no hay, las ventas no son posibles.
Alimento - Compra de alimento	-	Si las existencias son abundantes la compra debe disminuir; si son escasas la compra debe aumentar.
Alimento - Consumo de alimento	+	Si hay alimento, se puede consumir; si no hay alimento, tampoco hay consumo.
Capital - Egresos	+	Si hay capital se puede incurrir en egresos; si no hay dinero, no se pueden asumir egresos.
De sustantivo a verbo en diferente subsistema		
Animales - Compra de alimento	+	Si hay muchos animales, es necesario comprar más alimento; si hay pocos las compras de alimento necesarias son menores.
Animales - Consumo de alimento	+	Si hay muchos animales, consumen mucho alimento; si hay pocos, el consumo es menor.
Capital - Compra de animales	+	Si el capital es alto, se puede comprar una cantidad mayor de animales; si el capital es bajo las compras deben ser menores.
Capital - Compra de alimento	+	Si el capital es alto, se puede comprar una cantidad mayor de alimento; si el capital es bajo las compras deben ser menores.
De verbo a verbo en diferente subsistema		
Compra de animales – Egresos	+	La compra de animales ocasiona un egreso directamente proporcional.
Compra de alimento – Egresos	+	La compra de alimento ocasiona un egreso directamente proporcional.
Venta de animales – Ingreso	+	La venta de animales ocasiona un ingreso directamente proporcional.
Consumo de alimento - Venta de animales	+	El consumo de alimento determina la posibilidad de que se vendan los animales en forma directamente proporcional. Si no consumen suficiente, las ventas disminuyen.

Las relaciones que parten de un sustantivo y llegan a un verbo de otro subsistema, implican determinaciones o causalidades. Las que parten de un verbo hacia otro verbo, implican que un evento desencadena otro, ya sea en el mismo subsistema o en uno diferente. En todos estos casos los efectos son directa o inversamente proporcionales según su signo sea "+" o "-" respectivamente.

A partir del diagrama de influencias se desarrolla el de flujos y niveles. Los sustantivos se representan como niveles (rectángulos) y los verbos como flujos (llaves) de entrada o de salida del nivel, según las relaciones que van del verbo al sustantivo tengan signo "+" o "-" respectivamente. Luego se agregan al diagrama las demás relaciones. Se utilizan

abreviaturas para simplificar la legibilidad del diagrama y de las ecuaciones del modelo matemático.

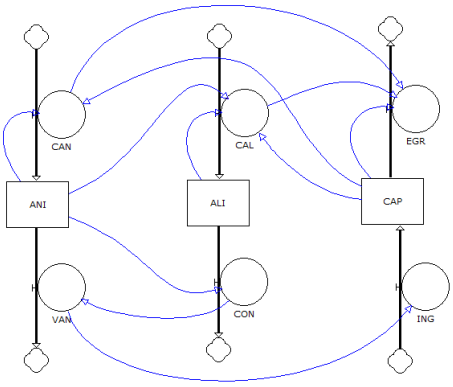

Figura 4 – Diagrama de flujos y niveles de la estructura básica del sistema

Modelamiento cuantitativo.

A cada nivel (sustantivo) le corresponde una ecuación diferencial. El valor futuro surge de sumar al valor actual los flujos de entrada y se restar los de salida.

Tabla 3 – Ecuaciones diferenciales de los niveles (sustantivos) del sistema

Sustantivo	Ecuación
Animales	$ANI(t+\Delta t) = ANI(t) + CAN(t) * \Delta t - VAN(t) * \Delta t$
Alimento	$ALI(t+\Delta t) = ALI(t) + CAL(t) * \Delta t - CON(t) * \Delta t$
Capital	$CAP(t+\Delta t) = CAP(t) + ING(t) * \Delta t - EGR(t) * \Delta t$

Por cada flujo (verbo) hay una ecuación auxiliar, construida únicamente con los elementos desde donde llegan influencias, y cuya validación de unidades lleva a los modeladores a identificar nuevos elementos del modelo. La única ecuación de flujo que se presenta en este artículo es la de compra de animales. Incluye animales y capital, pero la expresión *CAN(t)=f(ANI(t),CAP(t))* es problemática porque ningún operador permite obtener [i/t] a partir de [i] y [$]; hacen falta elementos.

Tabla 4 – Nuevos elementos del modelo

Nombre	Abreviatura	Tipo	Unidad	
Faltante de animales por comprar	FAC	Auxiliar	Individuo	[i]
Capacidad de compra de animales	CCA	Auxiliar	Individuo	[i]
Plazo de compra de animales	PCA	Parámetro	Tiempo	[t]

Es posible deducir que animales y capital no son la causa directa, sino que actúan a través de otros, mostrados en la Tabla 4, que le dan consistencia de unidades a la ecuación *CAN(t)=Mínimo(FAC(t),CCA(t))/PCA(t)*.

Modelamiento enriquecido del sistema.

En esta fase los nuevos elementos encontrados durante el modelamiento matemático, se incorporan al diagrama de influencias en un proceso iterativo, que produce un diagrama de influencias enriquecido y validado matemáticamente.

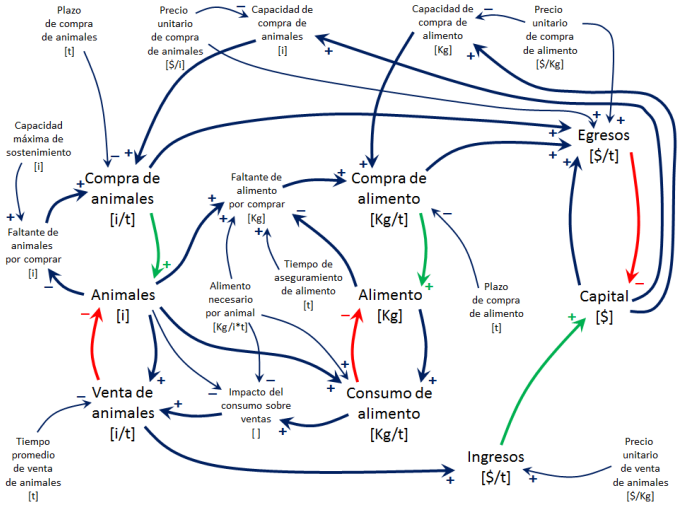

Figura 5 – Modelamiento cualitativo enriquecido del sistema

Todos los nuevos elementos se constituyen en requerimientos para el software. Los elementos que no reciben influencia de otros son parámetros, y se usan para configurar políticas de manejo del sistema, y realizar cálculos intermedios y las versiones finales de las ecuaciones de los flujos.

Tabla 5 – Parámetros del sistema

Nombre		**Unidad**
Plazo de compra de animales	PCA	[t]
Capacidad máxima de sostenimiento de animales	CMA	[i]
Precio unitario de compra de animales	PUA	[$/i]
Plazo de compra de alimento	PCL	[t]
Alimento necesario por animal	ANA	[Kg/(i*t)]
Tiempo de aseguramiento de alimento	TAL	[t]
Precio unitario de compra de alimento	PUL	[$/Kg]
Tiempo promedio de venta de animales	TVA	[t]
Precio unitario de venta de animales	PVA	[$/i]

Tabla 6 – Cálculos intermedios y ecuaciones de los flujos (verbos) del sistema

Elemento	Ecuación
Cálculos intermedios	
Faltante de animales por comprar	$FAC(t) = CMA(t) - ANI(t)$
Capacidad de compra de animales	$CCA(t) = CAP(t)/PUA(t)$
Faltante de alimento por comprar	$FLC(t) = (ANI(t) * ANA(t) * TAL(t)) - ALI(t)$
Capacidad de compra de alimento	$CCL(t) = CAP(t)/PUL(t)$
Impacto del consumo sobre las ventas	$ICV(t) = \sqrt{CON(t)/(ANI(t) * ANA(t))}$
Flujos del sistema	
Compra de animales	$CAN(t) = Mínimo(FAC(t), CCA(t))/PCA(t)$
Venta de animales	$VAN(t) = (ANI(t)/TVA(t)) * ICV(t)$
Compra de alimento	$CAL(t) = Mínimo(FLC(t), CCL(t))/PCL(t)$
Consumo de alimento	$CON(t) = Mínimo(ANI(t) * ANA(t), ALI(t))$
Egresos	$EGR(t) = CAN(t) * PUA(t) + CAL(t) * PUL(t)$
Ingresos	$ING(t) = VAN(t) * PVA(t)$

Transformación de modelos cualitativos y cuantitativos.

Para obtener un diagrama de clases: 1) los subsistemas se convierten en clases; 2) los sustantivos y los parámetros se transforman en atributos; 3) los verbos y cálculos auxiliares son métodos de clase. Para obtener un diagrama relacional: 1) cada subsistema se convierte en una tabla; 2) los sustantivos y parámetros se transforman en campos; 3) los cálculos auxiliares y verbos que requieran persistencia se convierten en tablas.

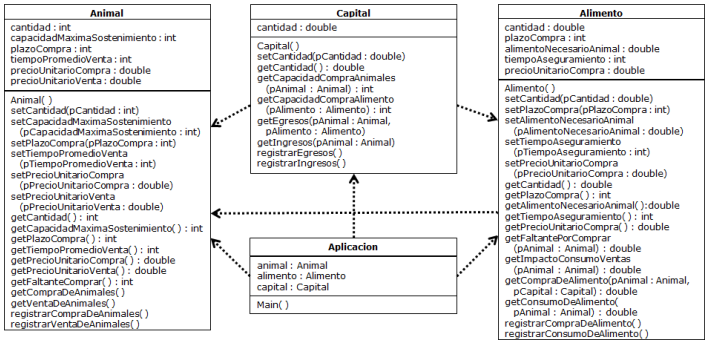

Figura 6 – Transformación del diagrama de influencias en diagrama de clases

Las ecuaciones que conforman el modelo matemático se utilizan para la implementación de los métodos de las clases. La verificación matemática de la

consistencia del modelo, y la inclusión de todos los parámetros y cálculos intermedios como requerimientos del software, asegura una mayor fiabilidad en las operaciones, que la que se obtiene modelando directamente el diagrama de clases.

```
67      public double getFaltantePorComprar(Animal pAnimal){
68          return (pAnimal.getCantidad()*alimentoNecesarioAnimal*tiempoAseguramiento)-cantidad;
69      }
70      public double getImpactoConsumoVentas(Animal pAnimal){
71          return Math.sqrt(getConsumoDeAlimento(pAnimal)/(pAnimal.getCantidad()*alimentoNecesarioAnimal));
72      }
73      public double getCompraDeAlimento(Animal pAnimal, Capital pCapital){
74          return Math.min(getFaltantePorComprar(pAnimal),pCapital.getCapacidadCompraAlimento(this))/plazoCompra;
75      }
76      public double getConsumoDeAlimento(Animal pAnimal){
77          return Math.min(pAnimal.getCantidad()*alimentoNecesarioAnimal, cantidad);
78      }
```

Figura 7 – Algunos métodos de la clase Alimento programados en lenguaje Java

5. Conclusiones

Viendo la Ingeniería de Requerimientos como aprendizaje sobre un problema, que conduce a la formulación de características y diseño de software, es factible utilizar DS para el modelamiento semántico en el proceso. Su carácter interdisciplinario e iterativo, centrado en la representación y mediado por un lenguaje no exclusivo de desarrolladores, permite abordar el conocimiento y las expectativas de los stakeholders, y llevarlos a diagramas que luego se transforman en diseño computacional, aprovechando la correspondencia entre pares semánticos: sustantivo-verbo, nivel-flujo, variable-derivada, atributo-método. El modelamiento matemático conduce al hallazgo de nuevos elementos, y enriquece tanto los requerimientos como los diseños y la programación del software.

Referencias bibliográficas

Abdel-Hamid, T. K. (1988). The Economics of Software Quality Assurance: A Simulation-Based Case Study. *MIS Quarterly, 12*(3), 395-411.

Andrade, H., Dyner, I., Espinosa, Á., López, H., & Sotaquirá, R. (2001). *Pensamiento Sistémico: diversidad en búsqueda de unidad* (1 ed.). Bucaramanga, Colombia: Universidad Industrial de Santander.

Aracil, J., & Gordillo, F. (1997). *Dinámica de Sistemas* (1 ed.). Madrid, España: Alianza Editorial.

Carlile, P. R. (2002). A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product. *Organization Science, 13*(4), 442-455.

Carlile, P. R. (2004). Transferring, Translating, and Transforming: An Integrative Framework for Managing. *Organization Science, 15*(5), 555-568.

Chang, L.-C., & Tu, Y.-M. (2005). Attempt to Integrate System Dynamics and UML in Business Process Modeling. *Proceedings of the 23rd International Conference of the System Dynamics Society*.

Charette, R. N. (2005). Why software fails. *IEEE Spectrum, 42*(9), 42-49.

Costain, G., & McKenna, B. (2012). Experiencing the elicitation of user requirements and recording them in use case diagrams through role-play. *Journal of Information Systems Education, 22*(4), 369-382.

Cox, K., Niazi, M., & Verner, J. (2009). Empirical study of Sommerville's and Sawyer's requirements engineering practices. *IET Software, 3*(5), 339-355.

Coyle, G. (2000). Qualitative and quantitative modelling in system dynamics: some research questions. *System Dynamics Review, 16*(3), 225-244.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31*(11), 1268-1287.

Date, C. J. (1998). Modelado semántico. En C. J. Date, *Introducción a los sistemas de bases de datos* (5 ed., Vol. 1, pág. 860). México DC, México: Addison Wesley Iberoamericana.

Forrester, J. (1994). System dynamics, systems thinking, and soft OR. *System Dynamics Review, 10*(2), 245-256.

Geels, F. W. (2007). Feelings of discontent and the promise of Middle Range Theory for STS: examples from technology dynamics. *Science, Technology & Human Values, 32*(6), 627-651.

Georgantzas, N. C., & Katsamakas, E. G. (2008). Information systems research with system dynamics. *System Dynamics Review, 24*(3), 247-264.

Gonzales, R. (2005). Developing the Requirements Discipline: Software vs. Systems. *IEEE Software, 22*(2), 59-61.

Größler, A. (2007). System dynamics projects that failed to make an impact. *System Dynamics Review, 23*(4), 437-452.

Hofmann, H., & Lehner, F. (2001). Requirements Engineering as a success factor in software projects. *IEEE Software, 18*(4), 58-66.

Hurtado, D. (2010). *Teoría general de sistemas. Un enfoque hacia la ingeniería de sistemas*. Barranquilla, Colombia: Fundación Universitaria San Martín.

Jackson, M. (2004). Seeing more of the world. *IEEE Software, 21*(6), 83-85.

Lawrence, B., Wiegers, K., & Ebert, C. (2001). The top risks of Requirements Engineering. *IEEE Software, 18*(6), 62-63.

Loucopoulos, P., & Preskas, N. (2003). A framework for Requirements Engineering using System Dynamics. *Proceedings of the 21st International Conference of the System Dynamics Society*.

Luna-Reyes, L. F., Black, L. J., Cresswell, A. M., & Pardo, T. A. (2008). Knowledge sharing and trust in collaborative requirements analysis. *System Dynamics Review, 24*(3), 265-297.

Madachy, R. J. (2008). *Software Process Dynamics*. Wiley Interscience - IEEE Press.

Napier, N., Mathiassen, L., & Johnson, R. (2005). Combining perceptions and prescriptions in Requirement Engineering process assesment: an industrial case study. *IEEE Transactions on Software Engineering, 35*(5), 593-606.

Nasir, M. H., & Sahibuddin, S. (2011). Critical success factors for software projects: A comparative study. *Scientific Research and Essays, 6*(10), 2174-2186.

Ramiller, N., & Wagner, E. (2012). Communication challenges in requirements definition: a classroom simulation. *Journal of Information Systems Education, 22*(4), 307-317.

Richardson, G. P. (2011). Reflections on the foundations of System Dynamics. *System Dynamics Review, 27*(3), 219-243.

Roberts, E. B. (2007). Making system dynamics useful: a personal memoir. *System Dynamics Review, 23*(2), 119-136.

Robertson, S. (2005). Learning from other disciplines. *IEEE Software, 22*(3), 54-56.

Sawyer, P., Rayson, P., & Cosh, K. (2005). Shallow knowledge as an aid to deep understanding in early phase Requierements Engineering. *IEEE Transactions on Software Engineering, 31*(11), 969-981.

Schwaninger, M., & Ríos, J. P. (2008). System Dynamics and Cybernetics: a synergetic pair. *System Dynamics Review, 24*(2), 145-174.

Sommerville, I. (2005). Integrated Requirements Engineering: a tutorial. *IEEE Software, 22*(1), 16-23.

Tang, R., & Huang, X. (2011). A software project management method based on trust and knowledge sharing. *Advanced Materials Research, 267*, 160-163.

Tignor, W. (2003). Stock and flow and UML relationships. *Proceedings of the 21st International Conference of the System Dynamics Society*.

Tignor, W. (2004). System Engineering and System Dynamics models. *Proceedings of the 22nd International Conference of the System Dynamics Society*.

Williams, D. (2001). Towards a system dynamics theory of requirements engineering process. *Proceedings of the 19th International Conference of the System Dynamics Society*.

Wolstenholme, E. (1999). Qualitative v. Quantitative Modeling: the evolving balance. *Journal of the Operational Research Society, 50*(4), 422-428.