Towards a Requirements Specification Multi-View Framework for Self-Adaptive Systems

Muñoz-Fenández Juan C, Gabriel Tamura, Raúl Mazo, Camille Salinesi

Towards a Requirements Specification Multi-View Framework for Self-Adaptive Systems

Juan C. Muñoz-Fernández*†, Gabriel Tamura*, Raúl Mazo† and Camille Salinesi†
*Facultad de Ingeniería, Universidad Icesi, Cali, Valle del Cauca, Colombia.
Email: {jcmunoz,gtamura}@icesi.edu.co
†CRI, Université Paris 1 Panthéon - Sorbonne, Paris, France.
Email: {raul.mazo,camille.salinesi}@univ-paris1.fr

***Abstract*–The research on requirements specification for self-adaptive systems has a growing interest in the academy and the industry. As a result, currently there exists different proposals for the specification of requirements for self-adaptive systems. Despite the momentum that this area has received in recent years, in the works reported in the literature we have identified shortcomings. We propose a new framework to represent the requirements of self-adaptive systems. This framework seeks to manage uncertainty and to be sufficiently expressive for self-adaptive systems, including the representation of all the relevant concepts. The concepts, represented in different views to be used in a 5-stage process. Specifically, we present: (i) a discussion of the challenges and problems encountered in the literature; (ii); our proposal to solve these challenges; and (iii) a case study of the problem and its application.**

Keywords: requirements engineering, requirements specification, adaptive systems, adaptation

I. Introducción

Los sistemas de software auto-adaptables son una aproximación prometedora para el manejo de la complejidad e incertidumbre en los sistemas de software dinámicos modernos [1]. Estos sistemas se adaptan a factores tanto internos como externos que cambian dinámicamente [2]. El cambio en dichos factores, puede requerir la alteración y evolución de los requisitos definidos en dichos sistemas en tiempo de ejecución. En contraste, los requisitos dentro del ciclo de vida del software clásico se consideran estáticos o por lo menos no modificables durante la ejecución del sistema.

En este contexto, la investigación en ingeniería de requisitos para sistemas auto-adaptables es un área que ha presentado avances importantes en la última década. Actualmente existen propuestas como las de Qureshi et al. [3] [4], Bencomo et a. [5], Song et al. [6] y Alférez et al. [7], para la definición de los requisitos en sistemas adaptables y auto-adaptables. Sin embargo, pese a los avances que se han logrado, no encontramos en nuestro análisis de la literatura un marco de trabajo para ingeniería de requisitos completamente apropiado y aplicable para los sistemas adaptables. Nosotros consideramos que un marco de trabajo apropiado debe: (i) permitir definir y relacionar las diferentes conceptos del sistema mediante modelos; (ii) representar claramente las relaciones entre las vistas de un meta-modelo común; (iii) manejar la incertidumbre asociada a estos sistemas; y (iv) ser aplicable en ambientes industriales.

La industria está cambiando, hoy en día más industrias manufactureras entregan productos complejos y altamente personalizados a sus clientes. Para afrontar esta complejidad y personalización, en esta industria se propusieron las líneas de productos (LP). Posteriormente estas LP fueron adaptadas a la industria de software y denominadas líneas de productos de software (LPS) [8]. Las LPS permiten generar múltiples productos finales aprovechando las características en común entre los productos, sin embargo no consideran los cambios en tiempo de ejecución. El soporte a cambios en tiempo de ejecución fue introducido con las líneas de productos de software dinámicas (LPSD) [9], como una evolución de las LPS. Las LPSD son útiles en el contexto de los sistemas de software adaptables, sin embargo se requieren más elementos para la ingeniería de requisitos en estos sistemas.

Muchas de las propuestas actuales para sistemas adaptables, se basan en el modelado por metas [10]. Este modelado representa un sistema por medio de metas, agentes, relaciones, entidades, acciones y restricciones. Una meta es un objetivo no operacional alcanzado por la interacción de múltiples actores del sistema [10], siendo de un nivel superior de abstracción, comparado con los requisitos. Entre las propuestas que utilizan modelado por metas se destacan KAOS [10], i* [11], Tropos [21], Adaptive RML [3] y la propuesta de Sawyer et al. [12]. Sin embargo, estas propuestas no incluyen todos los conceptos que consideramos importantes para la especificación de requisitos de sistemas auto-adaptables. Normalmente estos conceptos se agrupan en uno o más modelos y se utilizan a lo lago de un proceso.

El objetivo principal de este artículo es presentar nuestro avance hacia un marco de trabajo para la especificación de requisitos de sistemas auto-adaptables. Este marco se inspira en la propuesta de Sawyer et al. [12], siendo este un trabajo en desarrollo que requiere mayor experimentación para validar por ejemplo su expresividad. Para lograr este objetivo, presentamos nuestro caso de estudio en la Sección II. En la Sección III presentamos la sintaxis y semántica del lenguaje junto con el proceso de modelado del sistema en las distintas vistas propuestas. Posteriormente, aplicamos el proceso al caso de estudio en la Sección IV. En la Sección V se expone el trabajo relacionado en especificación de requisitos de sistemas adaptables. En la Sección VI exponemos, basados en la experiencia, algunas discusiones sobre problemas en el trabajo relacionado y nuestra agenda de investigación. Finalmente presentamos nuestras conclusiones en la Sección VII.

II. Caso de estudio

Nuestro caso de estudio de sistema de software auto-adaptable es un sistema de tiendas en línea desarrollada por componentes. En la tienda se ofrecen productos tangibles, que pueden ser ordenados y despachados físicamente a los clientes. El sistema de tiendas consta de los siguientes componentes: pagos, administración, ayuda, promoción, venta y envío. Para ejemplificar la propuesta, se restringió el modelado a los componentes relacionados con la promoción y venta de productos. Estos componentes tienen como meta principal permitir buscar, seleccionar y comprar productos acorde a las preferencias de los clientes. Adicionalmente, siempre que sea posible, el sistema debe cumplir con rápidos tiempos de respuesta, ser operativamente económico y tener un nivel adecuado de disponibilidad. Los componentes del sistema relevantes para el caso de estudio están representados en la Figura 1. En este caso de estudio consideramos que estos componentes ya están disponibles. Esta consideración se sustenta en que los sistemas grandes, con frecuencia reutilizan componentes existentes por ejemplo desde un repositorio de componentes de una LPS, evitando comenzar su desarrollo desde cero.

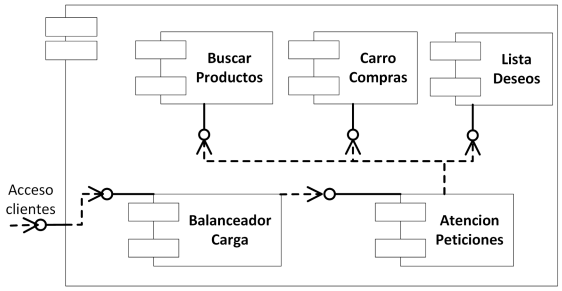

Figura 1: Componentes seleccionados para el caso de estudio

El componente *BalanceadorCarga* se incluyó para cumplir con los requisitos no funcionales: "rápidos tiempos de respuesta" y "alta disponibilidad". Este componente permite decidir, entre varios nodos de procesamiento disponibles, el nodo con mayor disponibilidad para atender la solicitud de un cliente. De esta forma se apunta a los dos requisitos enunciados previamente. Sin embargo, con un balanceo entre una cantidad fija de nodos no es posible, por ejemplo, controlar dinámicamente el costo de operación del sistema. Es necesario tener más información del sistema, su entorno y ejecución actual para tomar mejores decisiones y tener la capacidad de automatizar las mismas.

Sistemas como el presentado, requieren incluir variabilidad para controlar al menos parcialmente la incertidumbre en tiempo de ejecución por ejemplo, en cuanto a la impredecibilidad del número de usuarios que pueda tener una tienda en línea: mientras que en un día normal pueden haber 80 compradores al tiempo, en el denominado "back friday" esta cifra puede multiplicarse fácilmente al menos por 100. La incertidumbre que nos interesa afrontar se relaciona con los requerimientos no funcionales y la imposibilidad de conocer su nivel o posibilidad de satisfacción de antemano. Ante esta imposibilidad la incertidumbre puede afrontarse mejor si se incluye desde el modelado y diseño del sistema y no se limita únicamente a la implementación de los componentes. Debido a ésto, consideramos incluir la incertidumbre desde la elicitación de requerimientos y diseño en los sistemas auto-adaptables.

III. Lenguaje de modelado de requerimientos para sistemas adaptables

Nuestra propuesta para la ingeniería de requisitos en sistemas adaptables pretende ofrecer un marco de trabajo para la especificación de sistemas auto-adaptables. Los componentes de nuestra propuesta son: un meta-modelo común que define todos los conceptos y relaciones necesarios; la sintaxis y semántica de los conceptos y relaciones; un proceso de modelado con cinco fases; las vistas del meta-modelo necesarias para sistemas adaptables, que permiten instanciar los modelos; una ontología para la definición del contexto; y el soporte para el razonamiento y simulación de los sistemas resultantes. La definición del proceso de modelado y las vistas del meta-modelo se realizaron teniendo en cuenta nuestra experiencia previa en ingeniería de requisitos y en sistemas adaptables. Cada vista consta de conceptos, algunos de los cuales se pueden relacionar con conceptos de la misma y de otras vistas. Las vistas que proponemos se soportan sobre un meta-modelo común. Presentamos a continuación la sintaxis y semántica definida, el proceso de modelado y las vistas del meta-modelo propuestas.

III-A. Meta-modelo

Como base para formalizar y posteriormente implementar nuestro lenguaje, proponemos un meta-modelo que define los diferentes conceptos y relaciones del sistema. El meta-modelo es único incluyendo los diferentes tipos de conceptos como lo propone Guerra et al. [13] y se muestra en la Figura 2. Entre los conceptos más importantes se encuentran: afirmaciones de expertos, requerimientos no funcionales, dependencias no funcionales, activos, operacionalizaciones, requisitos funcionales y metas. Los activos incluyen los componentes de software.

El meta-modelo propuesto es el resultado de la agregación de cinco vistas. Por tanto, cada vista consta de un subconjunto de conceptos y relaciones del meta-modelo, como se ejemplifica para dos casos con las delimitaciones en la Figura 2. Cada vista del meta-modelo incluye adicionalmente las restricciones necesarias en OCL para obtener modelos correctos.

III-B. Sintaxis y semántica

A continuación presentamos la sintaxis y semántica de nuestro lenguaje. La sintaxis para un lenguaje visual, de acuerdo a Harel y Rumpe [14], se puede especificar en cuatro capas (segmentos, elementos geométricos, diagramas y condiciones de contexto) y cada capa se construye sobre los elementos definidos en la capa anterior. A continuación se detallan los elementos desde la segunda capa en adelante (la primera capa de Harel y Rumpe no se incluyó por ser muy básica y trivial).

En la segunda capa se definen los conceptos por medio de la combinación de rectángulos, trapecios y óvalos. Así mismo, las relaciones incluyendo direccionalidad por medio de flechas y arcos para representar restricciones de rango en las relaciones. Para los conceptos y relaciones se utiliza el color negro tanto para las líneas como para los textos. Adicionalmente, para el fondo de los conceptos se utiliza el color blanco y cuatro tonos diferentes de azul. Los colores permiten aumentar

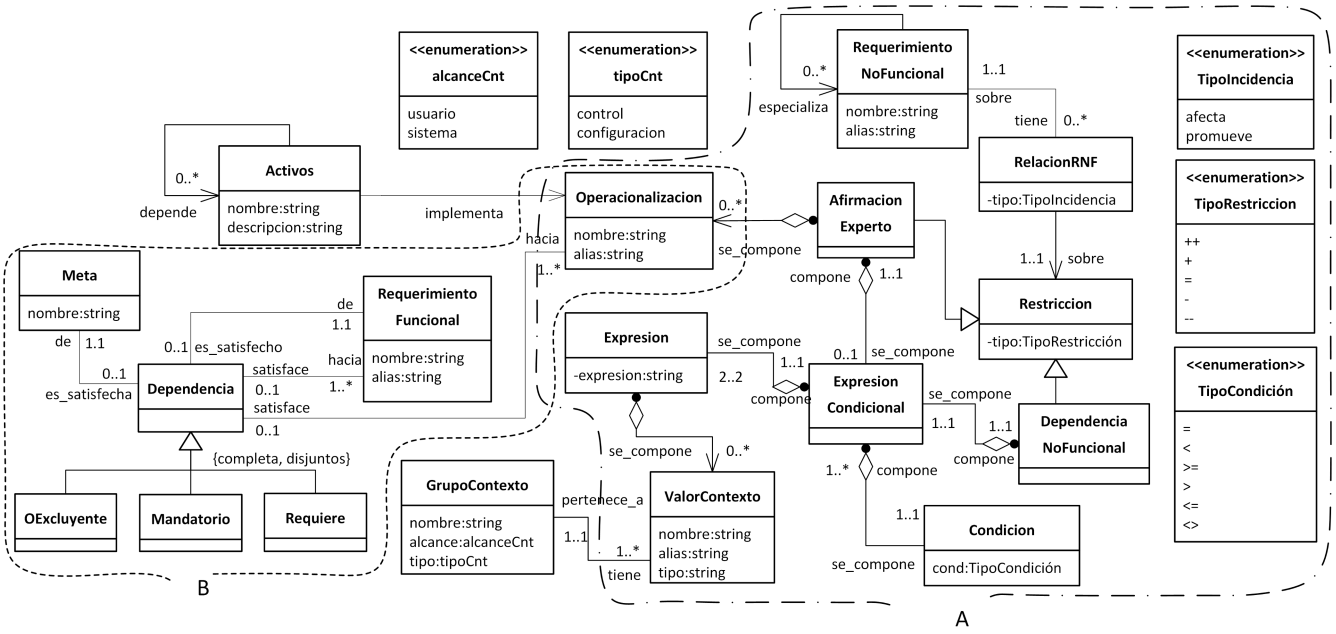

Figura 2: Meta-modelo para el lenguaje propuesto, con las vistas de afectación y niveles de satisfacción (A) y requisitos y variabilidad (B) señaladas

la separación semántica entre los conceptos, acorde a las recomendaciones de Moody [15] para los modelos. Estos cinco colores son correctamente identificables en impresión a blanco y negro. Los conceptos se caracterizan entonces por la combinación de sus formas y colores.

Para definir la funcionalidad y restricciones del sistema, se definieron cuatro conceptos. Estos conceptos son: metas, requisitos funcionales (RF), operacionalizaciones y requisitos no funcionales (RNF), que se representan en la Figura 3.

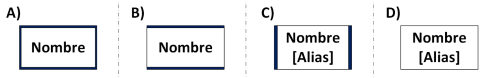

Figura 3: A) meta, B) requisito funcional, C) operacionalización, D) requisito no funcional.

a) Meta: Sintaxis: Nombre enmarcado en rectángulo con reborde azul oscuro en todo su contorno (Figura 3A). *Semántica*: Define un propósito funcional general del sistema, por ejemplo oferta y venta de productos en línea. El sistema debe velar por satisfacer en todo momento las metas definidas.

b) Requisito funcional: Sintaxis: Nombre enmarcado en rectángulo con reborde azul oscuro en los bordes superior e inferior (Figura 3B). *Semántica*: Un requisito funcional permite satisfacer parcial o totalmente una de las misiones del sistema. Los requisitos funcionales tienen por tanto un nivel de abstracción menor que las metas, por ejemplo buscar y comprar productos.

c) Operacionalización: Sintaxis: Nombre y alias que la identifica enmarcados en rectángulo con reborde azul oscuro en los bordes derecho e izquierdo (Figura 3C). *Semántica*: Una operacionalización especifica cómo se puede satisfacer parcial o totalmente un requisito funcional, por ejemplo nodo único de alta capacidad. Si las operacionalizaciones necesarias se satisfacen, acorde a la relación definida, el requisito funcional asociado a ellas se satisface.

d) Requisito no funcional (RNF): Sintaxis: Nombre y alias que lo identifica en rectángulo sin reborde azul (Figura 3). *Semántica*: Un requisito no funcional corresponde a restricciones necesarias sobre el sistema y no hace parte de su funcionalidad definida, por ejemplo garantizar un costo mínimo de operación. En general, el nivel de satisfacción óptimo en un RNF no puede ser siempre alcanzado, debido a esto se establece una escala o niveles de satisfacción parciales. Dependiendo del RNF, el nivel de satisfacción puede aplicar para el sistema en general o para un usuario particular. Los RNF se relacionan indirectamente con los RF, por medio de las restricciones que se asocian a las operacionalizaciones.

Dado que los mecanismos de adaptación en los sistemas auto-adaptables dependen del contexto, se definieron tres conceptos para representar el contexto por medio de variables. Estas variables deben ser monitoreadas en tiempo de ejecución. En la Figura 4 se muestran los grupos de variables, y las variables de sistema y de usuario.

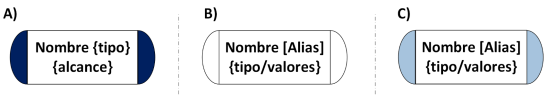

Figura 4: A) Grupo de contexto, B) variable de contexto global, C) variable de contexto local.

e) Grupo de contexto: Sintaxis: Nombre, tipo y alcance enmarcados en rectángulo ovalado, en ambos lados con fondo

azul oscuro (Figura 4A). *Semántica*: Un grupo de contexto agrupa las variables de contexto por alguna característica común, por ejemplo GCS1 control global. Un grupo de contexto puede incluir variables de control o configuración, dependiendo del tipo. Así mismo, dependiendo del alcance pueden ser locales o globales.

f) Variable global: Sintaxis: Nombre, alias que la identifica y el tipo de dato enmarcados en rectángulo ovalado, en ambos lados con fondo blanco (Figura 4B). *Semántica*: Una variable global es una representación abstraída de una variable, componente del sistema o su ambiente y que afectan al sistema en general, por ejemplo nivel de solicitudes por segundo. Para facilitar su utilización, su valor se puede obtener simplificando uno o más valores percibidos del sistema, de acuerdo a la necesidad de utilización.

g) Variable local: Sintaxis: Nombre, alias que la identifica y el tipo de dato enmarcados en rectángulo ovalado, en ambos lados con fondo azul claro (Figura 4B). *Semántica*: representa una variable o la instancia de un componente del sistema o su ambiente, cuyo alcance es local y por tanto la afectación es independiente para cada usuario del sistema, por ejemplo si el usuario está registrado o es un visitante (tipo de usuario). Se utilizan principalmente para los RNF cuyo satisfacción varía de forma independiente para cada usuario.

Los últimos tres conceptos de nuestra propuesta corresponden a las afirmaciones de expertos, dependencias no funcionales y los componentes de software, mostrados en la Figura 5.

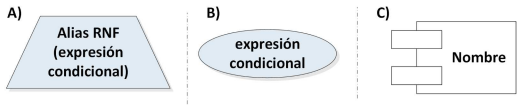

Figura 5: A) Afirmación de expertos, B) dependencia no funcional, C) componente.

h) Afirmación de expertos: Sintaxis: Alias de operacionalizaciones (en la parte superior) y/o una expresión condicional (en la parte inferior) enmarcados en trapecio de fondo azul muy claro (Figura 5A). Más de un alias de operacionalización, se separan por punto y coma. La expresión condicional, entre paréntesis, puede incluir combinación de valores, variables y operadores que da como resultado un valor lógico. *Semántica*: Define un supuesto sobre la combinación de operacionalizaciones y variables que pueden afectar el cumplimiento de los requisitos no funcionales, por ejemplo nodo único costo efectivo afecta el costo mínimo de operación. El tipo de afectación y el RNF afectado se representan posteriormente por medio de una relación.

i) Dependencia no funcional (DNF): Sintaxis: Expresión condicional enmarcado en óvalo de fondo azul muy claro (Figura 5B). La expresión condiciona, se define igual que en la afirmación de expertos. *Semántica*: Expresa una condición lógica que depende de las variables de contexto, cuando la condición se cumple se espera un nivel de satisfacción de un RNF, por ejemplo para un nivel del solicitudes por segundo inferior a la carga máxima de un nodo costo efectivo (NSS<CCE), se espera un nivel alto en la garantía de costo mínimo de operación. El nivel esperado y el RNF afectado se representan posteriormente por medio de una relación.

j) Componente: Sintaxis: nombre enmarcado en rectángulo con dos rectángulos pequeños adicionales sobre el borde izquierdo (Figura 5C). *Semántica*: Representa un componente de software del sistema, por ejemplo buscar productos. Los componentes clave representados son los que permiten implementar las operacionalizaciones.

El nivel de afectación, en las afirmaciones de experto y el nivel esperado, se representan con signos negativos, de igualdad o positivos (– –, –, =, +, + +).

Las relaciones se representan por medio de cinco tipos de líneas, mostradas en la Figura 6. Las relaciones en la Figura 6A y la Figura 6D, pueden incluir el nivel de satisfacción y esperanza respectivamente en ciertas relaciones, utilizando los modificadores definidos. La relación de la Figura 6C, incluye un arco con tres opciones para representar todos (“Y”), al menos uno (“O”) o solo uno (“O-Ex”). La semántica de las relaciones varía dependiendo de los conceptos relacionados y se explica en la siguiente capa para cada una de ellas.

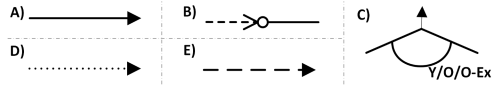

Figura 6: A) Relación entre afirmaciones de experto y un RNF, entre dos RNF y entre dos elementos de contexto, B) relación de uso entre dos componentes, C) Arcos entre una meta y varios RF, D) relación de una DNF con un RNF, E) relación de un componente con una operacionalización

En la tercera capa, se asocian los conceptos y las relaciones previamente definidos.

k) Relación medios para satisfacer un fin: Sintaxis: La relación se compone de un arco entre los orígenes de las relaciones (RF) y el destino (la meta), representado en la Figura 7A. En la parte inferior del arco se especifica la cardinalidad requerida para los RF. Esta relación aplica para dos casos adicionales: para la relación entre RF, mostrado en la Figura 7B; y para la relación entre RF y operacionalizaciones, mostrado en la Figura 7C. *Semántica*: Esta relación indica los conceptos orígenes que son requeridos para la satisfacción del concepto destino. Puede requerirse solo un concepto, por lo menos un concepto o todos, esto se define en el texto bajo el arco de la relación. En el caso de la Figura 7A, si se cumple que los requisitos funcionales necesarios se satisfacen, todos (“Y”), al menos uno (“O”) o uno y solo uno (“O-Ex”), la meta asociada a ellos también se satisface.

l) Relación de contexto: Sintaxis: Las relaciones entre variables de contexto y su grupo de contexto se representan con una flecha de línea sólida de la variable hacia el grupo de contexto al que pertenece, mostrada en la Figura 8A. *Semántica*: Esta relación representa el tipo de contexto al cual pertenece la variable.

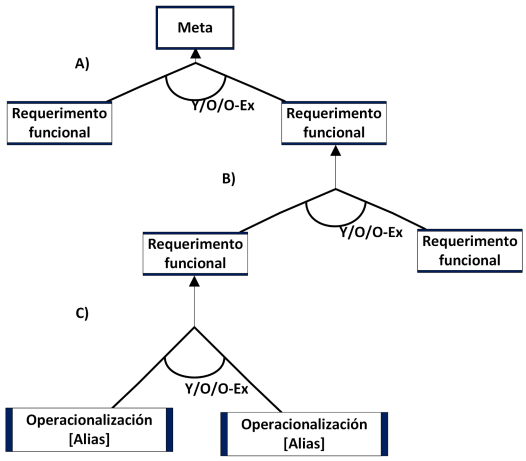

Figura 7: A) Relación entre meta y requisitos funcionales, B) relación entre requisitos funcionales, C) relación entre requisito funcional y operacionalizaciones.

m)Relación entre RNF: Sintaxis: La relación entre RNF se representa con una flecha de línea sólida con dirección del RNF más específico al RNF más general, mostrada en la Figura 8B. *Semántica*: Esta relación representa la especialización de los RNF.

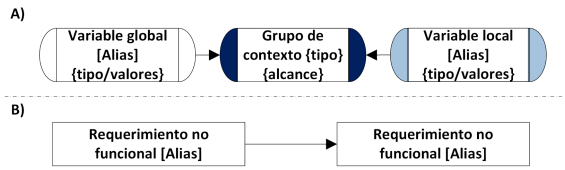

Figura 8: A)Relación entre elementos de contexto, B) relación entre requisitos no funcionales

n) Relación de afirmación de expertos: Sintaxis: La relación entre las afirmaciones de experto y los RNF se representan con una flecha de línea sólida de la afirmación hacia el RNF que afecta, mostrada en la Figura 9. En la relación se debe especificar el tipo de incidencia de las operacionalizaciones al relacionarlas (– –, –, =, +, + +). *Semántica*: Esta relación representa la afectación sobre un RNF por parte de una afirmación de expertos que puede incluir operacionalizaciones y una expresión condicional, como se describió en la sintaxis de la misma. La incidencia va desde muy negativa (– –), hasta muy positiva (+ +).

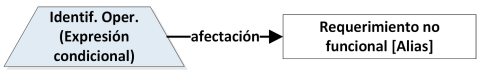

Figura 9: Relación de afirmación de expertos

ñ) Relación de dependencia no funcional: Sintaxis: La relación entre DNF y RNF se representan con una flecha punteada de la DNF hacia el RNF que afecta, mostrada en la Figura 10. En la relación se debe especificar el valor de satisfacción esperado del RNF relacionado. *Semántica*: Esta relación representa la esperanza sobre el cumplimiento de un RNF dada una dependencia no funcional que incluye una expresión condicional, como se describió en la sintaxis de la misma. Por ejemplo NSS<CCE (nivel del solicitudes por segundo inferior a la carga máxima de un nodo costo efectivo).

Figura 10: Relación de dependencia no funcional

o) Relación entre una operacionalización y un componente: Sintaxis: Se representa desde un componente hacia una operacionalización, por medio de una línea discontinua que no incluye texto, mostrada en la Figura 11. *Semántica*: Esta relación vincula una operacionalización con el componente adecuado para satisfacerla.

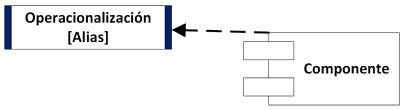

Figura 11: Relación entre operacionalizaciones y componentes

p) Relación entre componentes: Sintaxis: Se representa de un componente hacia otro, por medio de una línea discontinua que termina en punta y la recibe una línea continua que inicia en un círculo, mostrada en la Figura 12. *Semántica*: Representa el uso de un componente por parte de otro. Los componentes de software, se relacionan entre si para cumplir el objetivo del sistema mediante invocaciones de sus servicios.

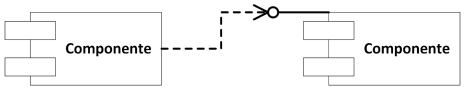

Figura 12: Relación entre componentes

En la cuarta capa, se definieron las siguientes condiciones de contexto para obtener diagramas correctos en el lenguaje propuesto:

- En las relaciones como medios para satisfacer un fin (Figura 7), el arco se debe crear entre al menos dos elementos relacionados.
- Los alias utilizados para las variables, operacionalizaciones y requisitos no funcionales deben ser únicos.
- Cada operacionalización o requisito funcional puede estar en una sola relación en calidad de concepto relacionado.
- Cada variable global o local debe estar asociada a un y solo un grupo de variables.
- Las afirmaciones de experto deben tener por lo menos una operacionalización o una condición de cumplimiento.
- Al evaluar las condiciones, se debe obtener un valor verdadero o falso.
- Todas las variables utilizadas en las expresiones debe estar definidas como variables globales o locales

- Todas las operacionalizaciones utilizadas en las afirmaciones de experto deben estar previamente definidas.
- Dos relaciones de afirmaciones de expertos, sobre las misma afirmación de expertos, no pueden satisfacerse al tiempo si tienen implicaciones diferentes sobre un mismo RNF.
- Dos dependencias no funcionales, no pueden satisfacerse al tiempo si tienen esperanza diferente sobre el mismo RNF.

III-C. Proceso de modelado

El proceso de modelado consta de cinco fases que se explican a continuación:

Fase 1: Definición de las metas y los requisitos funcionales (RF) y no funcionales (RNF). Inicialmente se definen la meta o metas de alto nivel que debe cumplir el sistema. Para cada una de ellas se deben definir los RF que la satisfacen individualmente o en conjunto (especificando solo uno ("O-Ex"), al menos uno ("O") o todos ("Y") en la relación). Los RF se pueden dividir en otros más específicos o alternativos entre ellos utilizando la misma notación mostrada en la Figura 7B. Una meta o RF puede tener un solo arco asociado a ella. Se deben definir también los requisitos no funcionales (RNF), en la Figura 3D se muestra su representación. Si un RNF es muy general es posible especializarlo en otros RNF, utilizando la notación de la Figura 8B. Se deben incluir los RNF que puedan verse afectados por cualquiera de los RF.

Fase 2: Definición de la variabilidad. Para cada RF, se definen las operacionalizaciones, de manera análoga a la definición de los RF para cada meta. Las operacionalizaciones posteriormente se enlazarán con componentes de software que implementen la funcionalidad esperada. La representación se muestra en la Figura 7C. Todas las operacionalizaciones que permiten satisfacer los RF se deben considerar. Sin embargo, se debe prestar especial atención a las operacionalizaciones que afectan positiva o negativamente los RNF. Si durante la definición de la variabilidad surgen nuevos RNF, estos se deben adicionar.

Fase 3: Variables de contexto. Se definen las variables globales y locales para el sistema tiendo en cuenta los conceptos definidos en las dos fases anteriores. Se incluyen las variables cuya alteración pueda afectar el nivel de cumplimiento de al menos un RNF. Esta afectación puede aplicar siempre o solo en el caso de seleccionar una o varias operacionalizaciones. Así mismo, se incluyen las variables cuya alteración afecte el nivel esperado de cumplimiento de al menos un RNF. Las variables pueden ser locales o globales, dependiendo del alcance requerido. Así mismo, pueden ser de configuración o control. El valor de las variables de configuración es definido por el administrador; variables de control deben ser monitoreadas por el sistema.

Fase 4: Afirmaciones de expertos y dependencias no funcionales. A continuación, sobre la variabilidad y el contexto establecidos, se definen las afirmaciones de expertos y se relacionan con cada uno de los RNF que evidencian afectación de las operacionalizaciones y variables. La selección de una o varias operacionalizaciones puede afectar positiva o negativamente un RNF, en algunos casos supeditado al cumplimiento de una expresión condicional. Así mismo, se definen las dependencias no funcionales con una expresión condicional y la relación de expresa la esperanza en la satisfacción del RNF, como se muestra en la Figura 10. La esperanza, por medio de la relación, se debe definir para todos los RNF que aplique.

Fase 5: Componentes y ensamblaje. En caso contar con un modelo previo de componentes, las relaciones entre estos componentes se debe revisar. Se deben hacer los cambios necesarios, identificando las relaciones que se deben quitar y agregar, y los componentes adicionales necesarios. En caso que el modelo no estuviera definido, se debe definir completamente. Finalmente, se asocian los componentes requeridos para cada una de las operacionalizaciones definidas en la fase 2.

III-D. Vistas del meta-modelo propuesto

Relacionado con el proceso previamente definido, nuestra propuesta define cinco vistas sobre el meta-modelo. Cada vista del meta-modelo determina un subconjunto de conceptos y relaciones tomados del meta-modelo común. En el meta-modelo presentado en la Figura 2, se delimitaron dos de las vistas. Las vistas del meta-modelo propuesto son las siguientes:

1) Requisitos y de variabilidad: En esta vista del sistema se representan las metas que se especializan en requisitos funcionales (RF) y estos RF a su vez se pueden descomponer en otros RF más específicos, definidos en la primera fase del proceso. Adicionalmente, incluye las operacionalizaciones definidas en la fase 2 del proceso, asociadas a los RF.

2) Requisitos no funcionales: Esta vista sirve para definir y relacionar los requisitos no funcionales del sistema (RNF), tanto generales como específicos. Estos requisitos se caracterizan porque su satisfacción se ve afectada por condiciones variantes del sistema o su entorno. Este modelo también se define en la fase 1 del proceso.

3) Contexto: Esta vista permite especificar la información de contexto del sistema por medio de variables. El modelo se representa formalmente a partir de un ontología que permite definir el alcance de las variables, los tipos, los valores o rangos permitidos para cada una de ellas. Este modelo toma las variables definidas en el fase 3 del proceso. Se pueden implementar ciclos re-alimentados, como lo propone Brun et al. [16], para el monitoreo y adaptación del sistema en respuesta a los cambios en las variables de control tanto nivel local como global.

4) Afectación y niveles de satisfacción: Un RNF se ve afectado por la operacionalizaciones escogidas para uno o varios RF y/o por una expresión condicional que incluye variables de contexto. Esta vista permite relacionar las afirmaciones de expertos y las dependencias no funcionales definidas en la fase 4 del proceso. Los conceptos y relaciones de esta vista se encuentran delimitados en la Figura 2, con la letra A.

Adicionalmente, se definieron tres restricciones invariables en OCL para que los modelos obtenidos sean correctos:

context AfirmacionExperto: inv self.tipo = afecta

context DependenciaNoFuncional:
inv self.tipo = promueve

context AfirmacionExperto: inv self-> operacionalizaciones->size() >0 or self.expresion <>null

Las restricciones invariables que tienen que ver con el tipo de incidencia se definen dado que una afirmación de experto conlleva una afectación (positiva o negativa) sobre un requerimiento no funcional. Una dependencia no funcional tiene una condición para la cual se espera un nivel de cumplimiento del requerimiento no funcional asociado, por lo tanto siempre promueve un RNF.

La última restricción garantiza que una afirmación de experto no se defina vacía. Debe tener una expresión condicional o por los menos una operacionalización para poder ser válida.

De esta misma forma se define la restricción invariante para garantizar que la relación dependencia de una meta solamente sea hacia requisitos funcionales; así como para las demás condiciones de contexto enunciadas.

5) Modelo de componentes de software y ensamblaje: Este modelo permite definir los componentes de software que representan las funcionalidades del sistema y sus relaciones. Este modelo se une solamente con el modelo de variabilidad por medio de relaciones que van desde los componentes hacia las operacionalizaciones definidas en la fase 5 del proceso. Para efectos de la simulación de nuestra propuesta, estos componentes se implementan como fachadas. El modelo de componentes se relaciona con las operacionalizaciones definidas previamente por medio de líneas punteadas. Estas representan la unión de los requisitos funcionales con la implementación, por medio de las operacionalizaciones definidas.

Con estos modelos buscamos alcanzar no solo una amplia dimensión del problema, sino también una adecuada comprensión de las diferentes dimensiones en la especificación de requisitos. Definimos también la relación entre el modelo de variabilidad y el modelo de requisitos no funcionales, por medio de restricciones, que soportan tanto las afirmaciones de los expertos como las dependencias no funcionales. Estas restricciones se implementan por medio una red de restricciones, pero su definición no se incluyó dentro del alcance de este artículo. Se incluye una clara separación de las metas, los requisitos funcionales y las operacionalizaciones. Adicionalmente, se incluye modelo de componentes de software que hace parte del dominio de la solución y se va modificando y completando gradualmente conforme avanza la comprensión del sistema.

IV. Aplicación del proceso al caso de estudio

En nuestro caso de estudio, se espera mantener en tiempo de ejecución tres requerimientos no funcionales. Teniendo en cuenta estos requerimientos, aplicamos el proceso anteriormente definido a la tienda en línea para los componentes relacionados con la oferta y venta de productos. Con ello se identificarán las operacionalizaciones y condiciones que pueden ayudar o afectar el cumplimiento de los requerimientos no funcionales definidos.

Fase 1: Definición de las metas y los requisitos funcionales y no funcionales. Acorde al proceso, para el caso de estudio se definió la meta "oferta y venta de productos en línea", que se observa en la Figura 13. Así mismo, se definieron tres requisitos no funcionales (RNF): alta disponibilidad (en respuesta a tener un nivel adecuado de disponibilidad); alto desempeño (para rápidos tiempos de respuesta); y costo mínimo (para ser operativamente económico).

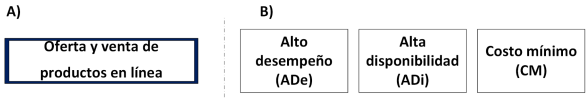

Figura 13: A) Meta y B) RNFs

Acorde a la meta se definieron tres requisitos funcionales (RFs), representados en el centro de la Figura 14. En este caso todos los requisitos son necesarios ("Y").

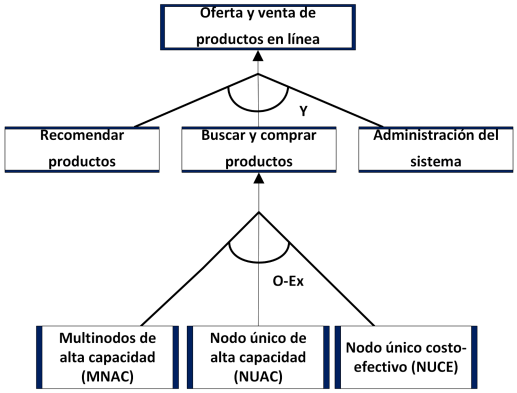

Figura 14: Meta (superior), RFs (centro) y operacionalizaciones (inferior)

Fase 2: Definición de la variabilidad. Para el RF *Buscar y comprar productos* se definieron tres operacionalizaciones que se observan en parte inferior de la Figura 14. Cualquiera de las operacionalizaciones independientemente permite satisfacer el RF, por lo cual el arco es "O excluyente" ("O-Ex"). Las tres operacionalizaciones son funcionalmente equivalentes, pero difieren en los RNF que ayudan a satisfacer. Las opciones disponibles corresponden a: un nodo que es económico pero no ofrece características de alta disponibilidad y rendimiento (nodo único costo-efectivo); un único nodo con características de alta disponibilidad y rendimiento; y múltiples nodos de alta disponibilidad y rendimiento.

Fase 3: Definición del modelo de contexto. Se definieron ocho variables de contexto, siete a nivel del sistema (global) y una a nivel de usuario (local) que se observan en la Figura 15. Para las variables NSS (Nivel de solicitudes por segundo), NN (Número de nodos), CCE (Carga Máxima nodo CE), CMN (Cantidad máxima) y CAD (Carga máxima nodo AD), se definió un valor entero. Para las variables NCP (Nivel carga promedio), T (Temporada) y TU (Tipo de Usuario) se definieron los valores en palabras relevantes de cada una que se observan en la Figura 15.

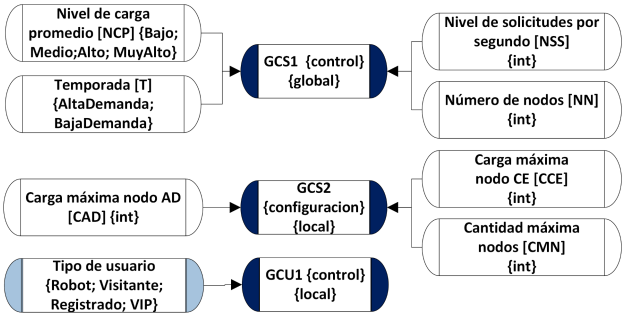

Figura 15: Variables de contexto

Fase 4: Afirmaciones de expertos y dependencias no funcionales. Se definieron las afirmaciones de expertos, una para cada RNF. En el caso del RNF *Costo mínimo*, se definieron cuatro afirmaciones de expertos, tres dependientes de operacionalizaciones y una dependiente de una expresión. Las afirmaciones de expertos para dicho RNF se muestran en la parte inferior derecha de la Figura 16.

Así mismo, para los tres RNF se definieron las dependencias no funcionales. En el caso del *Costo mínimo*, se espera cumplirlo sobre todo si el nivel del solicitudes es inferior a la carga máxima de un nodo costo efectivo (NSS<CCE); adicionalmente se espera un nivel moderado de costo cuando las solicitudes están entre carga máxima de un nodo costo efectivo y carga máxima de un nodo de alta disponibilidad (NSS>CCE && NSS<CAD). Por otro lado, el costo se impacta negativamente si el número de solicitudes por segundo es inferior al número de nodos menos uno por la carga máxima de un nodo de alta disponibilidad (NSS<(NdN-1)*CAD). De manera similar se definió para los otros dos RNF, como se muestra en la Figura 16.

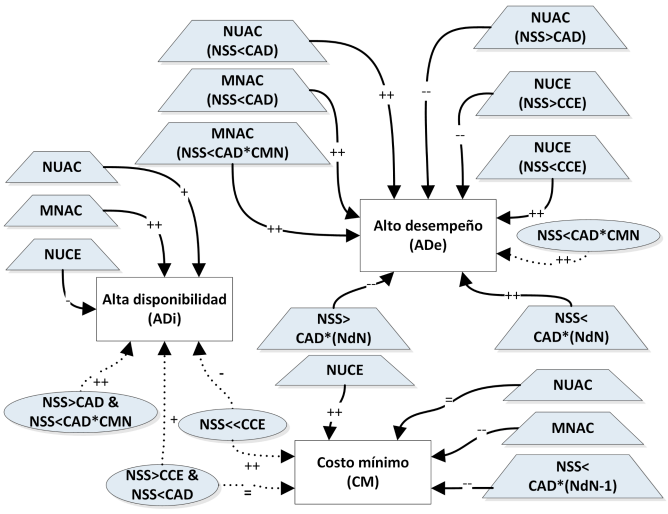

Figura 16: Afectación y niveles de satisfacción

Fase 5: Modelos de componentes y ensamblaje. Dado que ya se había incluido el componente de balanceo de carga, no se hicieron modificaciones al modelo de contexto previo. Para cada una de las operacionalizaciones, definimos el componente asociado a las misma. En la Figura 18, se muestra la operaciones multinodos de alta capacidad con el componente asociado. Para las dos operacionalizaciones restantes se realiza de manera similar.

Verificación de conformidad de los modelos: La conformidad de los modelos con las vistas del meta-modelo se verificó para los modelos de nuestro caso de estudio. A continuación se detalla la verificación de conformidad del modelo de afectación y niveles de satisfacción de la Figura 16 con la vista de afectación y niveles de satisfacción delimitada y marcada con la letra A en la Figura 2.

Las 16 afirmaciones de expertos son conformes con el concepto AfirmacionExperto. 12 de estas afirmaciones de experto tienen un alias de las tres operacionalizaciones definidas. Estas tres operacionealizaciones a su vez son conformes con el concepto Operacionalizacion. Adicionalmente, 9 de las afirmaciones de experto tienen expresiones condicionales. Las 9 expresiones condicionales son conformes con el concepto ExpresionCondicional y se constituye de una condición y dos expresiones. Las condiciones de las expresiones son conformes con el concepto Condicion y todas tienen uno tipo de condición válido. Finalmente, las dos expresiones de cada expresión condicional son confirmes con el concepto Expresion y están compuestas por uno o más alias de valores de contexto. Los valores de contexto son conformes con el concepto ValorContexto. La representación visual definida para las afirmaciones de experto solo muestra los alias, sin embargo en la Figura 17 se ejemplifica para una afirmación de experto la correspondencia de los alias a las operacionalizaciones y valores de contexto, esta ejemplificación no hace parte de nuestra sintaxis.

Las cuatro dependencias no funcionales son conformes con el concepto DependenciaNoFuncional. De manera análoga a las afirmaciones de experto, las dependencias no funcionales se componen de una expresión condicional con una condición y dos expresiones, conformes con los conceptos y relaciones del meta-modelo.

Por último los tres requerimientos no funcionales son conformes con el concepto RequerimientoNoFuncional.

Es fácil constatar que todas las representaciones de relaciones son conformes con las relaciones del meta-modelo.

Finalmente, también se cumple con las restricciones invariantes especificadas para las afirmaciones de experto y las dependencias no funcionales. Todas las afirmaciones de experto tienen líneas sólidas en las relaciones, las cuales representan la afectación sobre el RNF; todas las dependencias no funcionales tienen líneas punteadas en las relaciones, las cuales representan que promueven el RNF; y todas las afirmaciones de experto tienen por lo menos una operacionalización o una expresión condicional.

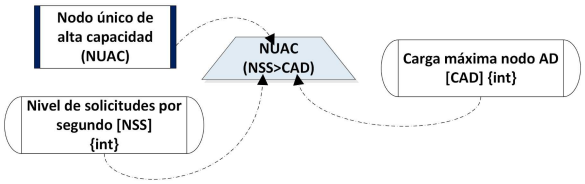

Figura 17: Ejemplificación de los alias en las afirmaciones de experto

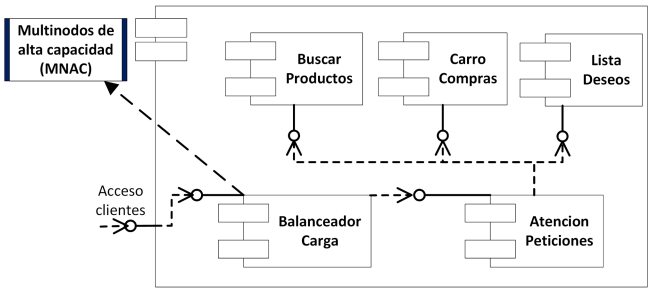

Figura 18: Componentes y ensamblaje

V. Trabajo relacionado

Para realizar la elicitación de requisitos para sistemas adaptables se han tenido diversas propuestas. Sin embargo, no hemos encontrado una que: (i) incluya todos los conceptos de las diferentes vistas del sistema, incluyendo sus relaciones; (ii) relacione los componentes de las vistas claramente; (iii) maneje la incertidumbre asociada a estos sistemas; y (iv) pueda ser aplicada directamente a ambientes industriales. Presentamos los cambios realizados al modelo de requisitos y de variabilidad, inspirado en la propuesta de Sawyer et al. [12] y otras propuestas existentes para sistemas adaptables.

V-A. Contraste con la propuesta de Sawyer et al.

Para nuestro trabajo, partimos del lenguaje de representación de requisitos propuesto por Sawyer et al. [12]. Este lenguaje propone una solución que integra el contexto, la variabilidad y los requisitos no funcionales en un solo modelo manteniendo la claridad en las relaciones. Sobre este lenguaje, proponemos varias modificaciones para extenderlo, enriquecerlo y aclararlo, dado que el mismo fue propuesto para un controlar inundaciones en la rivera de un rio por medio de sensores. Nuestra propuesta incluye las vistas de componentes y ensamblaje que no se encontraban en el lenguaje de referencia. Así mismo se extiende el manejo de la información de contexto y se enriquece la expresividad de los requisitos y la variabilidad. También se incluyen expresiones en las afirmaciones de expertos y dependencias no funcionales con modificaciones. Con ello se busca lograr un marco de trabajo más general y completo para los sistemas auto-adaptables. Las diferencias de nuestra propuesta con relación al modelo de Saywer et al. [12] se detallan a continuación:

a) Cardinalidades grupales: Se incluyeron alias para identificar los conceptos: RNF, variables y operacionalizaciones. Estos alias permiten reducir la cantidad de relaciones y utilizar más fácilmente los conceptos en las restricciones.

b) Cardinalidades grupales: Sawyer et al. [12] proponen utilizar "O-Ex" (XOR) entre las metas y su operacionalización y "Y" (AND) entre las metas y submetas; nosotros consideramos usar ambos tipos de arcos en los tres tipos de relaciones definidos. Con el cambio se reutilizan mejor las operacionalizaciones, al poder asociar más de una para satisfacer un requisitos funcional. En la Figura 14 se observa el arco "Y" para la meta *Oferta y venta de productos en línea* y "O-Ex' para *Buscar y comprar productos*.

c) Colores de fondo y figuras: La propuesta de Sawyer et al. [12] utiliza dos colores de fondo en los componentes del modelo, en contraste, nuestra propuesta utiliza cuatro colores de fondo, explicados en la Subsección III-B. Adicionalmente, varios conceptos se modificaron para aprovechar mejor el espacio.

d) Restricciones: Respecto a las restricciones se incluyeron varios cambios. Primero, nuestra propuesta utiliza el mismo color tanto para las dependencias no funcionales como para las afirmaciones de expertos. Esto se debe a que ambas son implementadas como restricciones, sobre diferentes tipos de conceptos. Segundo, en las relaciones de dependencias no funcionales y afirmaciones de expertos, se eliminó la relación con las variables de contexto y con las operacionalizaciones, respectivamente. Con ello el diagrama es más legible y el lenguaje más escalable. Tercero se incluyeron las expresiones tanto para las afirmaciones de expertos como para las dependencias no funcionales, aumentando la expresividad del lenguaje al aumentar el nivel de libertad para expresar las restricciones. Y cuarto, se incluyó la posibilidad de tener varias operacionalizaciones en una misma afirmación de expertos.

e) Relaciones en requisitos no funcionales: Las relaciones entre requisitos no funcionales y dependencias no funcionales propuestas por Sawyer et al. [12] utilizan líneas de guiones, en su lugar, nuestra propuesta utiliza líneas punteadas que mejoran la visibilidad de la relación, sobre todo en el caso de incidencia negativa. Con esta modificación, se evita que se pueda confundir la incidencia con las líneas. En la Figura 16 corresponden a las relaciones en la parte inferior.

Las mejoras anteriormente propuestas han tenido en cuenta la retroalimentación sobre el lenguaje de Sawyer et al. [12], por parte de estudiantes maestría para el modelado de sistemas en ejercicios y proyectos de clase.

V-B. Propuestas existentes

A continuación resaltamos aspectos importantes de otras propuestas encontradas relacionadas con la nuestra.

NFR Framework: Chung et al [17] proponen el marco de trabajo para RNF que maneja la incertidumbre en sistemas adaptables representando por medio de gráficos independientes de RNF. El marco de trabajo consta de un proceso de diseño que parte de los RNF, que son descompuestos e identifica las operacionalizaciones como alternativas. Posteriormente selecciona entre las operacionalizaciones, las justifica y finalmente evalúa el impacto de estas decisiones.

KAOS: KAOS (Knowledge Acquisition and autOmated Specification) fue propuesto por Dardenne et al. [10] y consta de: un modelo conceptual para la adquisición y estructuración de modelos de requisitos, tanto funcionales como no funcionales; las estrategias de adquisición de modelos de requisitos; y un asistente automatizado que provee guía sobre los procesos para las estrategias de adquisición. KAOS se considera una metodología, dado que provee tanto el lenguaje de modelado como el método para su utilización. KAOS se centra en los requisitos tardíos y solo parcialmente en la elicitación temprana y el diseño arquitectural. Los requisitos tardíos se enfocan en modelar el sistema junto con su ambiente [18].

*i**: Yu et al. proponen i* como un marco de trabajo para la elicitación de requisitos en las fases iniciales [11]. Este marco incluye los conceptos de metas, tareas, recursos y requisitos no funcionales, algunos tomados del marco de trabajo RNF. i* solamente se enfoca en la elicitación temprana y parte de la tardía. En i*, las metas pueden ser refinadas en metas más específicas, hasta llegar al nivel de tareas, sin embargo no es posible expresar si las metas o tareas son opcionales.

RELAX: es un lenguaje para manejar la incertidumbre en requisitos para sistemas auto-adaptables presentado por Whittle et al. [19]. La incertidumbre se representa por medio de requisitos cuyo cumplimiento depende de condiciones del ambiente o necesidades de cambio en los mismos requisitos. RELAX define un vocabulario con una sintaxis clara para enunciar dichos requisitos. El lenguaje propuesto es formalizado semánticamente por medio de una rama de la lógica temporal difusa.

Reflexión en los requisitos: Bencomo et al. proponen reflexión en los requisitos [5], apoyándose en RELAX y en una extensión del modelo de metas de KAOS. Esta propuesta busca que los requisitos se representen adecuadamente en tiempo de ejecución.

Tropos: es una metodología de desarrollo de software de requisitos que cubre desde la elicitación temprana hasta la implementación, pasando por la elicitación tardía, el diseño arquitectural y el diseño detallado [21], [22]. En la etapa de elicitación, se diferencian la elicitación temprana y tardía, con el mismo acercamiento conceptual y metodológico. En la elicitación temprana, Tropos permite definir las metas del sistema con los actores asociados. Posteriormente, en la elicitación tardía, el modelo conceptual puede ser extendido. Las metas diferencian entre las metas funcionales y los RNF.

Techne: es un lenguaje abstracto para el modelado de requisitos propuesto por Jureta et al. [23]. Techne se propone como un núcleo para la creación de nuevos lenguajes de requisitos en la fase de elicitación temprana y no define una sintaxis visual de modelado. En Techne, los requisitos no funcionales no son refinados y resueltos como tales, en su lugar, estos se representan por medio de una red de RF que constituyen una aproximación al requisito no funcional inicial.

Adaptive RML: propuesto por Qureshi et al. [3] modela y soporta la derivación en términos de requisitos adaptativos. Provee la sintaxis para diagramar la elicitación y representación de requisitos para sistemas auto-adaptables. Conserva la alineación con los lenguajes de modelado por metas y está mapeado a Techne. Adaptive RML permite especificar las metas y tareas obligatorias y opcionales por medio de relegación de relaciones, permitiendo establecer una alternativa cuando la primera opción no se puede satisfacer. La relegación se establece entre dos o más requisitos de tal forma que un requisito obligatorio pueda ser sustituido por uno opcional.

Souza et al. [24] proponen Awareness requirements (Aw-Reqs) siendo una nueva clase de requisitos que pueden monitorear el éxito o fracaso de otros requisitos. Los requisitos monitoreados pueden ser características no funcionales, así como metas, tareas o supuestos del dominio. Esta nueva clase de requisitos impone restricciones en el comportamiento de los requisitos monitoreados.

CARE: El marco de trabajo en ingeniería de requisitos adaptativos continuos, (CARE, por sus siglas en inglés) propuesto por Qureshi et al. [4], permite que el sistema realice refinamiento de los requisitos. CARE separa la fase de elicitación de la fase de diseño, ambas fases utilizan Techne para expresar las soluciones. CARE también define una ontología compartida que provee los conceptos para soportar el marco de trabajo.

Brun et al. [16] proponen la utilización de ciclos de re-alimentación (Feedback Loops) para los sistemas auto-adaptables. Vromant et al. [25] proponen una aplicación específica para un sistema adaptable de monitoreo de tráfico. La inclusión de ciclos MAPE en el modelo de contexto, hace parte de una extensión a nuestra propuesta actual.

Entre las propuestas que aprovechan el modelo por metas, también se encuentra la de Sawyer et al. [12]. Esta propuesta enfoca su aplicación en la simulación de un sistema físico de monitoreo por medio de sensores inalámbricos, denominado GridStix que ha sido usado para garantizar la seguridad en las riveras de los río Ribble y Dee en Inglaterra. La solución propuesta por Sawyer et al. [12] es apropiada en el contexto de este tipo de sistemas, sin embargo debe ser analizada, extendida y generalizada para aplicarla de manera general a sistemas dinámicos. Esta propuesta, con su caso de estudio, se encuentra formalizada e implementada por medio de una simulación, siendo tanto la formalización como la simulación soportadas por la programación por restricciones. Con la programación por restricciones es posible definir las funciones de optimización y hallar el conjunto de soluciones para el sistema dadas unas variables de contexto de entrada.

Song et al. proponen incluir las preferencias de usuario en la planificación de la adaptación en sistemas adaptables [6]. Las configuraciones y preferencias del usuario y las restricciones se resuelven como un problema de satisfacción de restricciones, en tiempo de ejecución. Esto permite hacer la verificación de las preferencias de los usuarios y la adaptación para satisfacer la mayor cantidad de metas.

Rainbow: Garlan et al [26] proponen la auto-adaptación basada en la arquitectura con una infraestructura reutilizable. Para ello proveen un ciclo realimentado que se comunica con el sistema objetivo, lo monitorea y ejecuta adaptaciones en tiempo de ejecución. Para permitir el reuso, se divide en cuatro capas: sistema, arquitectura, traducción y conocimiento de adaptación. Esta última es específica del sistema objetivo.

Por último, Alférez et al. proponen el uso de modelos de variabilidad para soportar la adaptación dinámica de la composición de servicios [7]. En su propuesta, para la conexión entre el modelo de variabilidad y las configuraciones utilizan programación por restricciones. Adicionalmente, consideran un modelo de contexto que permite formalizar y razonar sobre el conocimiento del contexto. Esta propuesta se enfoca específicamente en la composición de servicios y no tiene en cuenta los aspectos especiales de los requisitos no funcionales, ya que todos son abordados como características.

VI. Discusión y trabajo futuro

Uno de los aportes más importantes de nuestra propuesta son los conceptos y sus relaciones, soportadas por vistas de

un meta-modelo común. Una de las vistas que incorporamos es la de componentes de software y ensamblaje, que permite desde la elicitación de requisitos establecer el puente de los componentes con la implementación. En este sentido, nuestra propuesta no solo es útil para los requisitos tempranos sino también para los tardíos, a diferencia del caso de Techne [23]. También se diferencia de Techne respecto al modelo de requisitos no funcionales que no son aproximados a RF. Adicionalmente, Techne al ser un lenguaje abstracto para creación de lenguajes, no define ninguna representación visual a diferencia nuestro. De manera similar, RELAX [20] provee una sintaxis y semántica para la definición de lo requisitos bajo incertidumbre, sin entrar a especificar las vistas del sistema. Un ejemplo traducido sería, "El proceso de sincronización DEBE ejecutarse tantas veces como sea posible". Por otra parte, Rainbow [26] se concentra solamente en la arquitectura y no propone un modelamiento de los RF y RNF del sistema.

Nuestro modelo de contexto se define por medio de una ontología y solo las variables resultantes se incluyen en el modelo conjunto. Los valores de las variables se representan en las relaciones hacia las dependencias no funcionales, facilitando el entendimiento del modelo resultante. En el caso de Adaptive RML [3], la definición del contexto y su valor implica la inclusión de la definición de la variable en cada utilización, en el modelo de conceptos y relaciones propuesto, lo cual puede generar confusión.

Adicionalmente en nuestra propuesta incorporamos los conceptos de afirmaciones de expertos y dependencias no funcionales. Estos conceptos que brindan mayor expresividad de modelado y permiten identificar las operacionalizaciones más adecuadas acorde al contexto del sistema. Así mismo, estos conceptos permiten la evaluación de la satisfacción de los requisitos no funcionales. En nuestra propuesta, esta evaluación es posible realizarla en tiempo de ejecución. Con excepción de Sawyer et al. [12], las demás propuestas encontradas no incluyen estos conceptos para relacionar los requisitos no funcionales.

Nuestra propuesta cubre los conceptos definidos en i* resolviendo algunas de sus falencias. i* no permite definir soluciones alternativas; en contraste, en nuestra propuesta tanto la especialización de las metas como su operacionalización, permiten definir soluciones alternativas, por medio de la especificación de "O excluyente" ("O-Ex") en las relaciones de asociación de las metas, RF y las operacionalizaciones. Esto hace nuestra propuesta en este sentido más expresiva que i*. Por otro lado, el marco de trabajo RNF, relacionado con i*, plantea un proceso de diseño que puede complementar nuestras fases de modelado en lo que respecta a los RNF.

Como lo explicamos anteriormente, nuestra propuesta considera los aspectos necesarios para ser aplicable para la especificación de requisitos de sistemas dinámicos. Incorporamos el modelo de componentes de software y de contexto para tener una visión más completa del sistema a especificar. Con los cambios adicionales explicados previamente, consideramos que aumenta la expresividad y escalabilidad del modelo.

Song et al. proponen el uso de variables para las configuraciones y decisiones de los usuarios [6]. En este sentido, nuestra propuesta propone un tratamiento por medio de dos tipos de variables para mejorar la eficiencia del sistema. Clasificamos las variables en globales y locales. Las variables locales, aunque no están limitadas a preferencias de usuario, consideran los aspectos concernientes al contexto específico del usuario que no afectan el sistema de forma global. Sin embargo, nuestras variables se enfocan en modificaciones sobre los requisitos no funcionales, mientras que Song et al. [6] proponen modificar el peso relativo asignado a las metas, buscando primero la satisfacción de las más relevantes para el usuario.

Nuestra propuesta hace especial énfasis en los requisitos no funcionales, incluyendo las afirmaciones de expertos y las dependencias no funcionales. Otras propuestas no separan los requisitos [7], no permiten manejar los requisitos no funcionales como tales [23] o no consideran explícitamente los requisitos no funcionales [19].

Trabajo futuro

Nuestra propuesta no se basa en un lenguaje de modelado por metas existente; por el contrario, proponemos un nuevo lenguaje. Este nuevo lenguaje se soporta un meta-modelo único y no un conjunto de meta-modelos como sucede en otras propuestas encontradas en la literatura. Nuestro meta-modelo proporciona vistas para crear los modelos del sistema apoyado en los conceptos del meta-modelo. En consecuencia, este meta-modelo único soporta las cinco vistas del sistema explicadas en la Sección III-D. Este lenguaje tiene en cuenta la definición de las relaciones entre los conceptos de las diferentes vistas, por medio de restricciones. Hemos presentado la definición del meta-modelo, sin embargo las reglas de transformación para garantizar la consistencia entre diferentes vistas es un trabajo en desarrollo actualmente en nuestro grupo. Así mismo, es un trabajo en desarrollo, la ontología para soportar el modelo de contexto y la herramienta visual que permita modelar las diferentes vistas del sistema.

Adicional a la estructura, la sintaxis y la semántica en lenguaje natural, el marco de trabajo considera también la definición de la semántica y gramática formal del lenguaje de especificación. Ambas se desarrollarán posteriormente y sobre su definición gramatical basada en un meta-modelo se podrán definir las transformaciones e interacciones del sistema.

Una vez definida la formalización del sistema, será posible razonar sobre las diferentes configuraciones del mismo y comparar las alternativas cuantitativamente. Este razonamiento se hace por medio de las restricciones sobre el modelo. Por medio de la restricciones, es posible también implementar simulaciones para representar el sistema y sus respuestas ante los cambios en el contexto. Nuestra meta inicialmente es poder implementar la solución que permita razonar y simular sistemas como el propuesto en nuestro caso de estudio, basado en el marco de trabajo propuesto. Posteriormente, las soluciones deberán involucrar casos de aplicaciones industriales reales.

Finalmente, estamos conduciendo una revisión sistemática de la literatura en ingeniería de requisitos para sistemas auto-adaptables. Esta revisión nos permitirá identificar propuestas adicionales a las actualmente encontradas para esta clase de sistemas. Con esta identificación buscamos refinar y enriquecer tanto el meta-modelo que estamos desarrollando con las vistas del sistema que lo conforman y sus relaciones. Dentro del proceso de refinamiento, planeamos realizar evaluaciones formales y empíricas sobre la sintaxis concreta y la semántica presentadas en este artículo.

VII. Conclusiones

Hemos presentado el meta-modelo propuesto y explicado las vistas del mismo que componen nuestra propuesta con sus conceptos y relaciones. Para cada concepto y relación incluimos con su sintaxis concreta y su semántica. Así mismo, hemos explicado las fases definidas para el proceso de modelado y un ejemplo de aplicación del mismo. Dado que nuestra propuesta se inspira parcialmente en Sawyer et al. [12] explicamos las principales diferencias con dicha propuesta. Adicionalmente, hemos enunciado las propuestas existentes en el área de especificación de requisitos para sistemas adaptables y las hemos comparado con nuestra propuesta, identificando la agenda a seguir para nuestro marco de trabajo.

Nuestra propuesta, al no basarse en lenguajes de modelado por metas o meta-modelo existentes, no tiene las limitaciones propias de dichos lenguajes. Nuestro lenguaje permite una definición de los requisitos por medio de modelos que son conformes a las vistas del meta-modelo. Estas vistas permiten llegar a los modelos de contexto, de requisitos no funcionales, de requisitos funcionales, de componentes de software y de variabilidad para el sistema adaptable que se esté modelando. Estas vista se relacionan entre si al tener conceptos comunes a más de una de ellas. Con los RNF y las restricciones planteadas tanto como dependencias no funcionales como afirmaciones de experto buscamos afrontar la incertidumbre para los sistemas adaptables.

La propuesta plasmada constituye el punto de partida para nuestro trabajo hacia la realización del marco de trabajo enunciado para la ingeniería de requisitos para sistemas auto-adaptables que busca ser más completa que las existentes hasta el momento, con el objetivo final de lograr aplicarla en casos de industria.

Referencias

[1] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *FOSE*, L. C. Briand and A. L. Wolf, Eds., 2007, pp. 259–268.

[2] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. D. M. Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, "Software engineering for self-adaptive systems: A research roadmap," in *Software Engineering for Self-Adaptive Systems*, ser. Lecture Notes in Computer Science, B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds., vol. 5525. Springer, 2009, pp. 1–26.

[3] N. A. Qureshi, I. Jureta, and A. Perini, "Towards a requirements modeling language for self-adaptive systems," in *REFSQ*, ser. Lecture Notes in Computer Science, B. Regnell and D. E. Damian, Eds., vol. 7195. Springer, 2012, pp. 263–279.

[4] N. A. Qureshi, A. Perini, N. A. Ernst, and J. Mylopoulos, "Towards a continuous requirements engineering framework for self-adaptive systems," in *Requirements@Run.Time (RE@RunTime), 2010 First International Workshop on*, 2010, pp. 9–16.

[5] N. Bencomo, J. Whittle, P. Sawyer, A. Finkelstein, and E. Letier, "Requirements reflection: requirements as runtime entities," in *ICSE (2)*, J. Kramer, J. Bishop, P. T. Devanbu, and S. Uchitel, Eds. ACM, 2010, pp. 199–202.

[6] H. Song, S. Barrett, A. Clarke, and S. Clarke, "Self-adaptation with end-user preferences: Using run-time models and constraint solving," in *MoDELS*, ser. Lecture Notes in Computer Science, A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. J. Clarke, Eds., vol. 8107. Springer, 2013, pp. 555–571.

[7] G. H. Alférez, V. Pelechano, R. Mazo, C. Salinesi, and D. Diaz, "Dynamic adaptation of service compositions with variability models," *Journal of Systems and Software*, 2013.

[8] P. Clements and L. Northrop, "A framework for software product line practice," *SEI Interactive*, vol. 2, no. 3, 1999.

[9] S. O. Hallsteinsen, M. Hinchey, S. Park, and K. Schmid, "Dynamic software product lines," *IEEE Computer*, vol. 41, no. 4, pp. 93–95, 2008.

[10] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Sci. Comput. Program.*, vol. 20, no. 1-2, pp. 3–50, 1993.

[11] E. S. K. Yu, "Towards modeling and reasoning support for early-phase requirements engineering," in *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, ser. RE '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 226–. [Online]. Available: http://dl.acm.org/citation.cfm?id=827255.827807

[12] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, and D. Hughes, "Using constraint programming to manage configurations in self-adaptive systems," *IEEE Computer*, vol. 45, no. 10, pp. 56–63, 2012.

[13] E. Guerra, J. de Lara, A. Malizia, and P. Díaz, "Supporting user-oriented analysis for multi-view domain-specific visual languages," *Information & Software Technology*, vol. 51, no. 4, pp. 769–784, 2009.

[14] D. Harel and B. Rumpe, "Meaningful modeling: What's the semantics of "semantics"?" *IEEE Computer*, vol. 37, no. 10, pp. 64–72, 2004.

[15] D. L. Moody, "The "physics" of notations: a scientific approach to designing visual notations in software engineering," in *ICSE (2)*, J. Kramer, J. Bishop, P. T. Devanbu, and S. Uchitel, Eds. ACM, 2010, pp. 485–486.

[16] Y. Brun, G. D. M. Serugendo, C. Gacek, H. Giese, H. M. Kienle, M. Litoiu, H. A. Müller, M. Pezzè, and M. Shaw, "Engineering self-adaptive systems through feedback loops," in *Software Engineering for Self-Adaptive Systems*, 2009, pp. 48–70.

[17] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, ser. The Kluwer international series in software engineering. Kluwer Academic Publishers Group, Dordrecht, Netherlands, 1999.

[18] A. Lapouchnian, "Goal-Oriented Requirements Engineering: An Overview of the Current Research," Department of Computer Science, University of Toronto, Toronto, Canada, Tech. Rep., Juni 2005.

[19] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel, "Relax: a language to address uncertainty in self-adaptive systems requirement," *Requir. Eng.*, vol. 15, no. 2, pp. 177–196, 2010.

[20] N. A. Qureshi, I. Jureta, and A. Perini, "Requirements engineering for self-adaptive systems: Core ontology and problem statement," in *CAiSE*, ser. Lecture Notes in Computer Science, H. Mouratidis and C. Rolland, Eds., vol. 6741. Springer, 2011, pp. 33–47.

[21] F. Giunchiglia, J. Odell, and G. Weiß, Eds., *Agent-Oriented Software Engineering III, Third International Workshop, AOSE 2002, Bologna, Italy, July 15, 2002, Revised Papers and Invited Contributions*, ser. Lecture Notes in Computer Science, vol. 2585. Springer, 2003.

[22] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.

[23] I. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *RE*. IEEE Computer Society, 2010, pp. 115–124.

[24] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness Requirements for Adaptive Systems," in *Proc. of the 6$^{\text{th}}$ International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2011, pp. 60–69.

[25] P. Vromant, D. Weyns, S. Malek, and J. Andersson, "On interacting control loops in self-adaptive systems," in *SEAMS*, H. Giese and B. H. C. Cheng, Eds. ACM, 2011, pp. 202–207.

[26] D. Garlan, S.-W. Cheng, A.-C. Huang, B. R. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *IEEE Computer*, vol. 37, no. 10, pp. 46–54, 2004.