Herramienta para la Compensación de Parámetros de QoS y Seguridad

Ana Nieto, Javier Lopez
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga, España
Email:{nieto,jlm}@lcc.uma.es

***Resumen*—El análisis conjunto de mecanismos de seguridad y QoS es esencial para las redes heterogéneas donde diversos dispositivos pueden coexistir en entornos dinámicos. En concreto, los dispositivos no siempre pueden ser conocidos, por lo que diferentes requisitos y mecanismos pueden surgir para el análisis. En este artículo, proponemos una herramienta para facilitar la configuración de entornos basada en el análisis paramétrico de dependencias, tomando como base de conocimiento un conjunto de parámetros de seguridad y QoS. Esta forma de análisis de parámetros a alto nivel permite considerar las dependencias y la compensación entre mecanismos con independencia del sistema de información subyacente. Posibilita por tanto evaluar el impacto que tales mecanismos, y otros definidos acorde al modelo, tienen sobre un sistema previo a su despliegue.**

***Palabras clave*—CPRM; QoS; PRM; Seguridad;**

I. Introducción y Fundamentos

Diversos modelos para el análisis conjunto de aspectos de seguridad y calidad de servicio (QoS) emergen como consecuencia directa de la amplia diversidad de dispositivos que componen las redes heterogéneas. En particular, los modelos genéricos para el análisis del balanceo o compensación de requisitos de seguridad y QoS son, desde el punto de vista práctico, los más relevantes para las redes heterogéneas de composición dinámica, en las que no se puede prever con gran exactitud los dispositivos que formarán la red.

Definimos un modelo genérico para el análisis de la compensación de seguridad y QoS como aquel que se abstrae de detalles específicos de una tecnología y que ofrece la posibilidad de integrar en el estudio cualquier tipo de tecnología y dispositivo a distinto nivel. De hecho, podemos encontrar algunos ejemplos de modelos genéricos en la literatura que se ajustan en mayor o menor medida a esta definición [1], [2], enfoques más específicos sobre seguridad o QoS [8], [4], [5], [3], y otros, que emplean técnicas paramétricas para mejorar la configuración de servicios [10]. Por ejemplo, en [1] se emplean técnicas de *model checking* para verificar las equivalencias entre especificaciones de seguridad y QoS, con el objetivo de controlar los flujos de información ilegítimos en el sistema. No obstante, obliga a definir un modelo de comunicación entre las aplicaciones del sistema, restringiendo por tanto su ámbito de uso. Alternativamente, en [2] se define un modelo basado en el contexto, que proporciona una función de utilidad para tener en consideración las preferencias del usuario. Sin embargo, no permite medir el impacto que unos parámetros del sistema tienen sobre otros, y el conjunto de contextos es limitado.

No obstante, el análisis conjunto de los mecanismos de seguridad y QoS debería basarse en el estudio de relaciones paramétricas, es decir, relaciones de dependencia entre los parámetros que definen la composición de los mecanismos de seguridad y los de QoS. Además, definir estas relaciones en base a un contexto es básico para expresar la relevancia de los parámetros, relaciones, operaciones y otros componentes y propiedades, que tienen cabida en el sistema de información.

I-A. Definición de un Modelo para el Análisis de Relaciones Paramétricas basado en el Contexto (CPRM)

En base al paradigma actual y futura convergencia de las redes, en [6] definimos un modelo para estudiar las relaciones paramétricas basado en el contexto, denominado CPRM por sus siglas en inglés (*Context-based Parametric Relationship Model*). Dicho modelo define la estructura de un sistema en base a un conjunto de parámetros y sus relaciones, un conjunto de operaciones que definen efectos sobre los parámetros dependientes, y una estructura de pesos que define la relevancia subjetiva y no subjetiva de los componentes del modelo.

Por ejemplo, un administrador puede considerar subjetivamente que la confianza es un parámetro clave para la subsistencia del sistema de información. En ese caso, el parámetro confianza tendría un peso mayor en el sistema que otros parámetros menos relevantes dado el caso. A su vez, los mecanismos que implementen el valor de confianza podrían heredar la relevancia o peso de su parámetro padre, en este caso, el parámetro confianza. Estos valores subjetivos estarían sujetos a la variabilidad del contexto, de forma que en un momento dado, ya sea por las medidas de seguridad adoptadas o por el entorno donde está el individuo, su relevancia puede variar. Por ejemplo, en un entorno familiar bien definido, el parámetro confianza y los mecanismos estrechamente dependientes podrían relajar su relevancia de no existir otras dependencias que se lo impidan. Esto es así, porque en el contexto *hogar* el individuo podría asumir que la confianza viene dada por su ubicación. Aunque no tiene porqué ser así.

Además, el modelo también contempla valores no subjetivos; destinados a definir el impacto o reacción en cadena que podría ocasionar una dependencia. Estos valores, se definen, en primer lugar, de forma aproximada en las dependencias del contexto general (GC, *General Context*), mientras que, una vez que los parámetros son instanciados, el peso es actualizado al contexto particular (PC, *Particular Context*).

ISBN: 978-84-9717-323-0

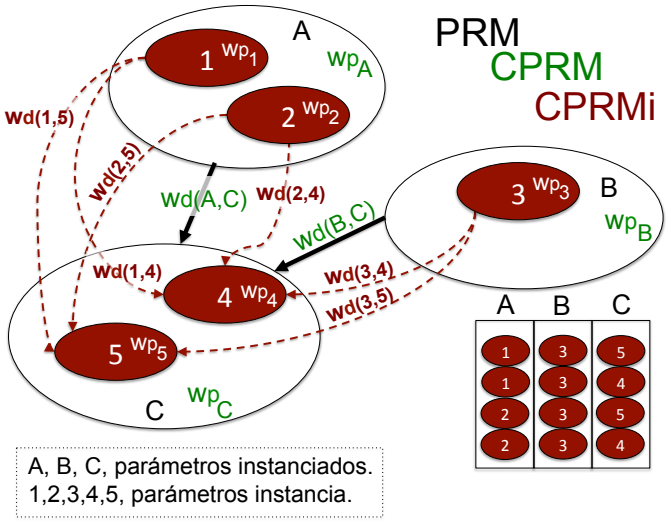


Figura 1. Instanciación de Parámetros.

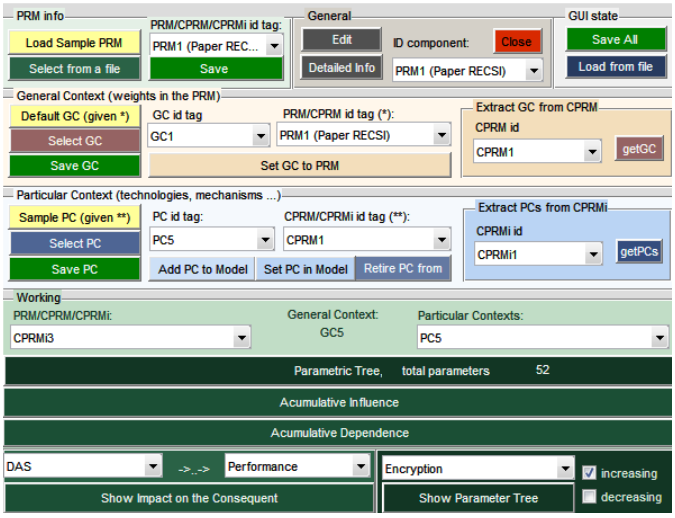


Figura 2. Interfaz de Administración.

La Figura 1 muestra parte de un sistema de dependencias paramétrico instanciado siguiendo el modelo CPRM dada su definición en [6][1]. La descripción de la formulación matemática asociada a las dependencias, así como las reglas de coherencia para la integración de contextos pueden consultarse en trabajos previos con más detalle [7]. En este caso, nos centramos en la visión general de cómo la integración de los parámetros y su instanciación quedarían reflejadas.

Partimos de un conjunto de parámetros que definen, de forma general, el escenario a evaluar. Este contexto base (BC) es fijo y no varía, y se encuentra en el PRM [2]. Pueden variar los pesos/relevancia de los parámetros, pero el BC siempre queda presente a la espera de que sus parámetros, relaciones, niveles, tipos y operaciones tomen los valores de contexto definidos en el GC y, posteriormente, en los sucesivos PCs. El BC es el resultado de un proceso de análisis exhaustivo sobre las arquitecturas y el entorno donde la herramienta tendrá cabida. En nuestro caso, surge del estudio de mecanismos de Seguridad y QoS en el *Internet del Futuro* [7]. Aunque la herramienta propuesta permite definir BC personalizados, siendo por tanto extensible a otros ámbitos de estudio, nuestro principal objetivo es su uso para el análisis de la compensación entre parámetros de Seguridad y QoS. En efecto, el BC que proporcionamos define dichos tipos de relaciones y no otros, que deberían ser agregados con posterioridad, según el caso.

En este artículo proporcionamos las directrices básicas para la implementación y el uso del modelo por medio de una herramienta desarrollada a tal efecto, a la que denominaremos SQT, por sus siglas en inglés *Security and QoS tradeoff Tool*. SQT proporciona un interfaz gráfico para la administración (Figura I-A) permitiendo al operador importar esquemas de modelo (PRM, CPRM, $CPRM_i$) y de contexto (GCs y PCs), salvar cualquier esquema en ficheros para su posterior uso y modificación, así como el espacio de trabajo completo, con los modelos y contextos asociados. También es posible extraer o eliminar contextos de los esquemas de modelo contextuales (CPRM, $CPRM_i$). El objetivo final es el análisis dirigido a la obtención de mediciones sobre el modelo de dependencias:

1. Incremento y decremento de parámetros.
2. Selección de conjuntos de parámetros por tipo y nivel.
3. Selección de parámetros instanciados (llamados padre) o bien de sus instancias (llamadas hijos) para distinguir entre diferentes opciones de configuración.
4. Calcular árboles de dependencias específicos para un parámetro, con el fin de posibilitar un examen más exhaustivo sobre el proceso de incremento/decremento.

A su vez, SQT permite visionar los resultados mediante diagramas de barras superpuestas que indican el impacto de un conjunto de parámetros en el resto de parámetros del sistema, o bien sobre un conjunto específico, en base al tipo/nivel, etc. También es posible seleccionar un parámetro en particular, como veremos en el caso de estudio. Otro modo de representación empleado es el uso de grafos, por medio de GraphViz. Así, el modelo de dependencias es representado mediante un grafo, en el que los parámetros se muestran acorde con la representación del tipo y agrupados por niveles según se define en el modelo.

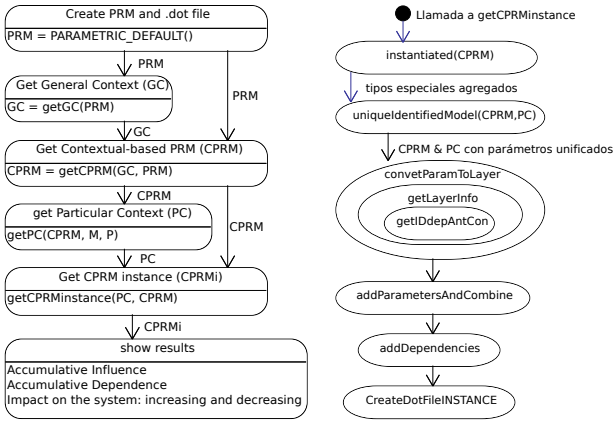
Para posibilitar el cumplimiento de tales requisitos, la herramienta implementa un conjunto de reglas de coherencia definidas para el modelo en [6]. El cumplimiento de estas reglas garantiza que el sistema paramétrico final mantiene la coherencia entre las dependencias.

El resto del artículo se divide como sigue. La Sección II muestra los detalles de implementación del prototipo conforme los requisitos dados. La Sección III estudia la usabilidad del prototipo para el análisis de la compensación entre parámetros de Seguridad y QoS. Por último, exponemos las conclusiones

[1]Hacemos referencia al modelo que define las estructuras PRM, CPRM, $CPRM_i$ (modelo instanciado a partir de un CPRM) y la relación entre sus componentes como CPRM.

[2]De cara a nuestro estudio, el BC no representa una estructura contextual, ya que los parámetros en el BC (en el PRM) carecen de pesos.

y el trabajo futuro.



(a) Generación $CPRMi$. (b) Integración de un PC.

Figura 3. Modelo de Componentes.

II. Prototipado del Modelo

El prototipo del modelo CPRM fue implementado en Matlab, ofreciendo una versión *plug-in*. Para el uso de SQT con toda su funcionalidad, debe instalarse GraphViz, a fin de interpretar los ficheros .dot que contienen las dependencias [3]. Los siguientes apartados abordan el diseño de SQT.

II-A. Modelo de Componentes

El diseño de SQT está basado en el uso de componentes, tanto desde el punto de vista arquitectural, como desde el punto de vista de la integración de contextos, considerados como componentes intercambiables. Así, teniendo en cuenta que en un CPRM puede existir un único GC, cualquier estructura de modelo puede ser ampliada/modificada usando SQT por medio de la agregación/sustitución de un GC, y de tantos PCs como sea preciso. Cuando en el CPRM se integra un PC, decimos que se genera una instancia del CPRM y lo denotamos como $CPRM_i$.

Así mismo, de cara a facilitar la tarea de análisis, es preciso que podamos volver a una versión anterior del modelo retirando el último contexto agregado, o construir nuevos contextos retirando alguno de los contextos integrados (no necesariamente el último). Esto es posible gracias a las reglas de integración y coherencia definidas para el modelo que implementa SQT [6], y a la definición de la cadena de integración y estructuras de datos descritas aquí.

Para permitir dicha funcionalidad, la integración de componentes se efectúa en SQT conforme al diagrama de actividad mostrado en la Figura 3(a), en el que se ilustra la creación de un $CPRM_i$ a partir de un PRM[4], para un caso de prueba. Para ello, creamos estructuras intermedias por defecto.

En particular, empleamos las funciones getGC y getPC para extraer o generar un contexto en base a una estructura paramétrica. En el caso de getGC, si la estructura introducida es un PRM, y, por consiguiente, sin contexto asociado, generará un GC asociado a los parámetros de la estructura. Si, por el contrario, recibe un CPRM o un $CPRM_i$, que son estructuras con un GC asociado, entonces devolverá el GC asociado a la estructura. De igual forma, getPC sólo devuelve los PC asociados a una estructura cuando están definidos, es decir, cuando la entrada es un $CPRM_i$. En otro caso, devolvería un PC aleatorio adecuado al tipo de estructura[5].

Por otra parte, las funciones getCPRM y getCPRMinstance asocian contextos con modelos. Es decir, getCPRM recibe un modelo y un GC que asignará al modelo. De esta asignación se obtiene un modelo paramétrico contextualizado (CPRM) coherente. El caso de getCPRMinstance es ligeramente distinto, ya que en un $CPRM_i$ varios PCs pueden coexistir. Aunque ambas funciones persiguen obtener una estructura nueva a partir de un modelo y un contexto, en el caso de getCPRMi se precisa un análisis mucho más exhaustivo.

Dado que un $CPRM_i$ es una instancia de un CPRM, se espera que sea una estructura dinámica, donde los PCs son intercambiados con mucha más frecuencia que un GC, que, aunque puede ser modificado, se asume que es una parte mucho más estable. Por tanto, el caso en el que diferentes parámetros se identifiquen igual en un $CPRM_i$ y un PC podría ser posible, dada la diversidad de escenarios que podrían definirse como PC. Estos casos deben considerarse para no solapar comportamientos de distintos parámetros. Este es sólo un ejemplo de los aspectos a contemplar en la integración de PCs en un $CPRM_i$.

La Figura 3(b) muestra de forma más detallada la secuencia de acciones realizadas por la función de integración de PCs, getCPRMinstance. Esta función recibe una estructura CPRM y la transforma en el primer paso para incluir los tipos y campos adicionales en una estructura $CPRM_i$. Si la estructura ya es un $CPRM_i$, entonces no realiza ningún cambio inicial, se considera que la estructura está instanciada.

El siguiente paso, es asegurarnos de que los identificadores de los parámetros en el modelo, ya un $CPRM_i$, no coinciden con los identificadores de los parámetros del PC. Tras este paso, obtenemos un modelo unificado, en el que los parámetros del modelo y el contexto significan lo mismo. Para estudiar la compensación paramétrica de los parámetros instanciados, getCPRMinstance convierte en niveles los parámetros padre, es decir, aquellos parámetros p para los que existen parámetros en PC que instancian a p (de forma matemática: $p|\exists p2 \in PC, p \in P(p2)$). Estos niveles contienen información de interés para, en caso de retirar el PC que provocó la instanciación, que el nivel asociado a un padre desaparezca y se restaure como parámetro sin instanciar.

Por último, se establecerán las dependencias que heredará el hijo, y se crearán aquellas necesarias para mantener el modelo

[3]Los ficheros .dot pueden ser modificados directamente o interpretados desde otras herramientas.

[4]En este ejemplo, el PRM es creado usando una función por defecto acorde a la definición del modelo.

[5]M y P indican el número de parámetros instancia que queremos que se generen por cada parámetro del modelo que se recibe como entrada.

coherente. Por ejemplo, si una instancia (hijo) se relaciona con un parámetro con el que el parámetro padre no tiene relación, se agregaría una nueva dependencia entre el parámetro padre y el parámetro con el que se relaciona el hijo (ej. Figura 1).

Finalmente, a nivel de análisis, todas las pruebas posibles sobre un PRM son posibles sobre estructuras CPRM o $CPRM_i$ (inclusive la interpretación mediante diagramas .dot). La diferencia sustancial, es que mientras que un PRM es estático, un CPRM presenta también una visión subjetiva del contexto de la red, dando más relevancia a unos parámetros, relaciones u operaciones conforme a las prioridades de administración o el conocimiento profundo de la red. Un $CPRM_i$, además, permite la integración de partes dinámicas en base a particularidades, contextos más variables y fugaces, pero también más específicos. Una vez que se conocen el conjunto de dispositivos tanto como para establecer sus dependencias y darles valores no subjetivos, sino próximos a la realidad, partes del GC pueden ser instanciadas con el PC.

II-B. Estructuras de Datos

Aunque en los ejemplos anteriores se mostró la creación de GCs y PCs por defecto en base a un modelo o estructura, cualquier estructura PRM, CPRM, $CPRM_i$, PC o GC tiene su formato predefinido con el que son creadas y empleadas.

La herramienta mantiene todas estas estructuras como parte de una estructura general, S, que gestiona los esquemas y contextos y que puede ser salvada, como espacio de trabajo.

Desde el punto de vista de la implementación, se puede considerar que S (Exp. 1) contiene el modelo de datos, compuesto por las estructuras de modelo y de contexto. En particular, la Tabla I muestra, grosso modo, las diferencias existentes entre los esquemas de modelo. Éstas, permiten identificar cuándo una estructura de modelo es un PRM, un CPRM ó un $CPRM_i$, y gestionar las operaciones definidas acorde al tipo de estructura y su definición. Las partes comunes entre los modelos, son las que posibilitan la integración basada en componentes. En particular, como parte del esquema PRM, las propiedades de niveles, tipos, operaciones y parámetros contienen elementos comunes como por ejemplo identificadores inequívocos, nombre (*string*), y forma de representación visual en los diagramas Matlab o GraphViz (color y forma). Además, cada parámetro, una vez calculadas sus dependencias con el resto, conserva la matriz de dependencias paramétrica, definida en [7], creada de forma recursiva, que define todas las relaciones de dependencia posibles que involucran al parámetro. Dichas matrices ocupan espacio en la estructura del PRM, a cambio de evitar el cálculo de mapas repetidas veces. Se obtienen a su vez de la matriz de dependencias general, donde se muestran todas las relaciones simples en su forma matricial binaria. Este conocimiento se extrae a su vez de las denominadas dependencias en bruto (DB), que expresan las relaciones $A \rightarrow B$ por medio del identificador del parámetro A, el de la operación de dependencia y el del parámetro B.

$$S = \{D1, D2, D3, D4, D5\}; \quad (1)$$
$$D1 = \#prm, nxtID, \{\{prm1, id, info, file\}, ...\}; \quad (2)$$
$$D2 = \#cprm, nxtID, \{\{cprm1, id, info, gcid, file\}, ...\}; \quad (3)$$
$$D3 = \#cprm_i, nxtID, \{\{cpmi1, id, info, gcid, pclist, file\}, ...\}; \quad (4)$$
$$pclist = [pcid1, pcid2, ...]; \quad (5)$$
$$D4 = \#gc, nxtID, \{\{gc1, gcid, info, file\}, ...\}; \quad (6)$$
$$D5 = \#pc, nxtID, \{\{pc1, gcid1, info, file\}, ...\}; \quad (7)$$

El esquema para el PRM sienta las bases de los esquemas definidos para el modelo CPRM y las instancias $CPRM_i$. No obstante, hay diferencias que, aunque sutiles en el esquema, suponen un cambio notorio en el proceso de cálculo de SQT. Así, mientras que la estructura CPRM supone un punto de inflexión entre un PRM y un modelo instanciado, los cambios realmente relevantes se producen de cara a la definición de un $CPRM_i$. Esto se debe en gran medida a dos factores clave: la definición de los tipos especiales para las instancias de parámetros y los parámetros instanciados, y la conversión puntual de parámetros como niveles. Estos factores, junto a la capacidad de restauración y modificación del modelo por medio de la eliminación y agregación de contextos, suponen un gran cambio respecto los modelos no instanciados, que quedan relegados a un desempeño más estático.

II-B1. Estructuras para los Contextos: A modo de ejemplo, mostramos a continuación dos esbozos de definiciones de GC (Exp. 8-14) y PC (Exp.15-17).

$$GC(1, 1:2) = \{NL\{id\_nivel1\ peso1;\ id\_nivel2\ peso2; ...\}\} \quad (8)$$
$$GC(2, 1:2) = \{NT\{id\_tipo1\ peso1;\ id\_tipo2\ peso2; ...\}\} \quad (9)$$
$$GC(3, 1:2) = \{NO\{id\_op1\ peso1;\ id\_op2\ peso2; ...\}\} \quad (10)$$
$$GC(4) = \{\}; \quad (11)$$
$$GC(5, 1:2) = \{NP, NProp\}; \quad (12)$$
$$GC(6:(5+NP), 1:NProp) = \{id\_param1\ peso1; ...\} \quad (13)$$
$$GC(6+NP, 1:2) = \{ND, \{id\_dep1\ peso1; ...\}\} \quad (14)$$

Las estructuras de contexto comparten algunos campos con las estructuras de modelo. Esto es preciso dado que las primeras pretenden efectuar cambios sobre los componentes de los modelos (parámetros, relaciones, tipos...). No obstante, los campos NProp y NP hacen referencia a la propia estructura de contexto, no a los campos del modelo. Es decir, las estructuras de contexto definen su propia forma de extensión. Por ejemplo, en la versión actual, la parte de definición de parámetros en un PC cuenta con 5 campos de propiedad (NProp=5): una lista de identificadores de parámetros padre (idPadres), el identificador del parámetro (que puede ser modificado si las reglas de integración lo demandan), el nombre del parámetro y el peso.

$$PC(1, 1:4) = \{NP, Nprop, ND, \{IDpc, descrip.\}\}; \quad (15)$$
$$PC(2:(1+NP), 1:Nprop) = \{idPadres, id, nombre, peso\}; \quad (16)$$
$$PC\{3+NP\} = \{idParamA, idOp, idParamB, peso; ...\}; \quad (17)$$

Dado un PC, cuando un CPRM es instanciado (Fig. 3(b)), se crean dos tipos especiales: *instance* e *instantiated*. Así, cuando

Tabla I
CAMPOS PARA LAS ESTRUCTURAS DE DATOS DE LOS ESQUEMAS DE MODELOS

Fila,Columna: Propósito	Definición PRM	CPRM (cambios sobre PRM)	$CPRM_i$ (cambios sobre CPRM)
1,1-2: Info. niveles	Número de niveles (NL) + Propiedades de Niveles	Agrega a las propiedades de cada nivel un peso w_l	Define niveles especiales para los parámetros instanciados
2,1-2: Info. tipos	Número de tipos (NT) + Propiedades de Tipos	Agrega a las propiedades de cada tipo un peso w_t	Agrega los dos tipos especiales: *instance* e *instantiated*
3,1-2: Info. operaciones	Número de operaciones (NO) + Propiedades de Operaciones	Agrega a las propiedades de cada operación un peso w_o	-
4,1-2: Otra información	Directorio por defecto (DD)	-	Agrega información isobre las instancias realizadas
5,1: NP	Propio del modelo		
5,2: NProp	5	6	6
5+NP,1-NProp: Parámetros	Propiedades de Parámetros	Agrega a las propiedades de cada parámetro un peso w_p	Los parámetros instanciados cambian su nivel por el nuevo creado como resultado de su instanciación
6+NP,1: Dependencias	Dependencias en bruto (DB) ó Matriz de dependencias procesada (MD)		
6+NP,2-3: Tras procesar DB	matriz de ceros NPxNP + DB	Matriz de costes NPxNP + DB	-

un parámetro sea instanciado y se cree un nivel a partir de éste, se etiquetará al parámetro como *instantiated* permitiendo aplicar las reglas de herencia para el cálculo del impacto paramédico. A su vez, cuando el parámetro es etiquetado como *instance*, se espera un identificador del PC que provocó la instanciación, y se tiene en cuenta que el parámetro es más dinámico que un parámetro que no sea instancia.

III. CASO DE USO Y EVALUACIÓN

En esta sección mostraremos cómo realizar pruebas para estimar la compensación entre requisitos de Seguridad y QoS.

III-A. Parámetros del Contexto Base

El ejemplo propuesto para el caso de análisis está basado en el funcionamiento de una red de sensores. Como tal, considera como parte del conjunto de parámetros del contexto base (BC) aquellos parámetros generales que pueden estar relacionados con una red de sensores, así como las relaciones entre éstos (consultar [7]). Adaptado al caso que nos ocupa, los parámetros del BC son mostrados en la Tabla II[6].

Aunque el GC por defecto para estos parámetros es inicialmente establecido con peso igual a 1 para todos los parámetros ($\forall p | p \in PRM, w_p = 1$), es posible establecer un GC subjetivo, basado en nuestras prioridades de administración. Por ejemplo, aumentar la relevancia/impacto del cifrado (*Encryption*), de tal forma, que todos los parámetros que tengan una dependencia en la que *Encryption* se encuentre en el antecedente serán más afectados que el resto de parámetros. Los parámetros afectados por el incremento del parámetro *Encryption*, pueden consultarse usando el árbol paramétrico particularizado para un parámetro. El efecto, sin embargo, podrá variar dependiendo del tipo de relación definida entre los parámetros y de los pesos definidos para las relaciones. Por ahora, todos los pesos para las relaciones tienen valor unitario ($w_d = 1, \forall d : A \rightarrow B | d \in PRM$). Estos pesos pueden modificarse con un GC, pero en nuestro caso lo haremos con un ejemplo de instanciación de parámetros.

[6]Las dependencias entre los parámetros no son mostradas debido a su extensión. Puede consultarse el diagrama ampliado que contiene estos y otros parámetros en [7].

Tabla II
PARÁMETROS DEL CONTEXTO BASE (BC)

	HIGH-LEVEL REQUIREMENTS
QoS	Reliability, Fault Tolerance, Availability
Security	Authentication, Authorization, Confidentiality, Integrity, Trust, Privacy
	LOCAL PROPERTIES
Resources	PowerConsumption, Memory, Rayleigh Channel, Energy, ComputationTime
Security	Anti-Tampering, Encryption, Public Key Cryptography, Symmetric Cryptography, Secure Key Exchange, Secure Key redistribution, Key Generation, Signature Scheme
	COMMUNICATION
QoS	Data Rate, Packet Size, Signal Strength, Data Transmission, Transmission Time, Transmission Power
Characteristics	Time-sleeping, Required-time-on
Consequence	Retransmmission
	MEASUREMENTS
QoS	Throughput, Delay, Jitter, Packet Loss, Response Time, Bit Error Rate (BER)
	ENVIRONMENT
QoS	Allowable Bandwidth, Error Probability
Attacks	DoS, Malicious Devices
Consequence	Interference, Congestion, Overhead, Fading, Shadowing, Noise

III-B. Agregación de un Contexto

Una vez aplicado el GC, podemos aplicar diferentes PCs sobre el CPRM resultante. Este hecho conduce a lo que denominamos *instanciación del modelo paramétrico*. A modo de ejemplo, mostraremos los cambios producidos en el sistema al aplicar el contexto particular mostrado en la Tabla III, cuyos pesos son estimaciones acorde al trabajo [9].

Tabla III
PESOS w_d CONFORME [9]

	Dependence			Weight
General Parameter	Antecedent	R	Consequent	w_d
Authentication	CAS	+	ECDSA	1
	DAS	+	ECDSA	1
	CAS	$\neg c$	Memory	0
	DAS	$\neg c$	Memory	5
	CAS	c	PacketSize	5
	DAS	c	PacketSize	1
Signature Scheme	ECDSA	$\neg c$	Energy	1
	PairingBased	$\neg c$	Energy	5
	ECDSA	c	Computation Time	1
	PairingBased	c	ComputationTime	5

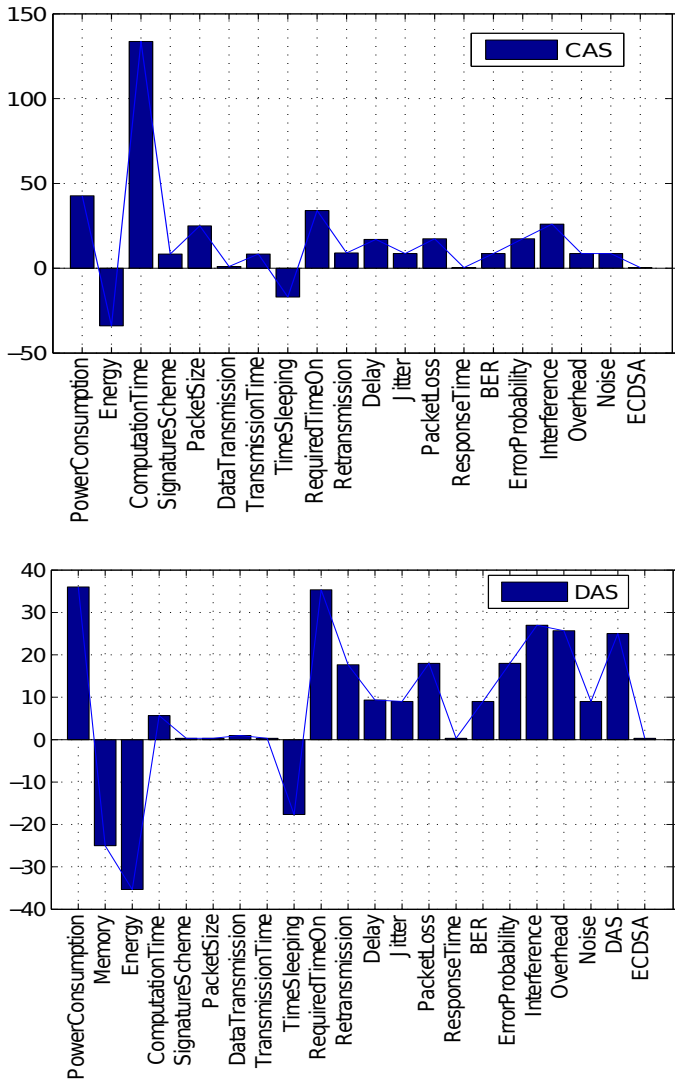


Figura 4. Impacto de CAS y DAS sobre el Rendimiento.

La Tabla III muestra los parámetros generales que serán instanciados (Authentication y SignatureScheme) y los parámetros instancia (CAS, DAS, ECDSA, PairingBased). Una vez integrado el nuevo contexto, los parámetros de Authentication y SignatureSheme pasarían a ser niveles, y como tales pueden ser consultados. Esta es una ventaja adicional del modelo, ya que permite comprobar el efecto que este último cambio de contexto tuvo sobre parámetros que ya se encontraban en el modelo. En el nuevo contexto final, cada vez que se incremente el parámetro Authentication o SignatureScheme, también serán incrementados los parámetros instancia, y con ellos los parámetros dependientes de éstos, que han podido introducir nuevas dependencias para hacer el modelo coherente.

Finalmente, el proceso de ajuste entre parámetros de Seguridad y QoS, se realiza en base al BC definido y la instanciación del modelo con los mecanismos cuyo impacto en el sistema resultante queremos medir. Por ejemplo, una vez introducido el último contexto (Tabla III), podemos evaluar el impacto que los mecanismos de autenticación CAS y DAS tienen sobre los parámetros de rendimiento (Figura 4) o de cualquier otro tipo. Note que los parámetros sobre los que se percibe el efecto no fueron obtenidos de [9], sino que son resultado de la integración con el BC definido a priori. La información del sistema será mucho más fiable y enriquecedora conforme el número de PC integrados sea mayor.

IV. Conclusiones y trabajo futuro

En este artículo proporcionamos las directrices básicas para la implementación y el uso de una herramienta para la evalucación de la compensación entre parámetros de Seguridad y QoS (SQT). SQT está basada en un modelo genérico para la compensación paramétrica basado en el contexto (CPRM) definido en trabajos previos. Un requisito importante perseguido es que cualquier contexto pueda ser intercambiado en un CPRM por otro nuevo o modificado. El caso de estudio abordado muestra cómo es posible emplear SQT para los fines propuestos.

Agradecimientos

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Economía y Competitividad a través del proyecto ARES (CSD2007-00004). Adicionalmente, ha sido financiado por la Junta de Andalucía través del proyecto FISICCO (TIC-07223). El primer autor ha sido subvencionado por el Programa FPI.

Referencias

[1] Alessandro Aldini and Marco Bernardo. A formal approach to the integrated analysis of security and qos. *Reliability Engineering & System Safety*, 92(11):1503–1520, 2007.

[2] Mourad Alia, Marc Lacoste, Ruan He, and Frank Eliassen. Putting together qos and security in autonomic pervasive systems. In *Proceedings of the 6th ACM workshop on QoS and security for wireless and mobile networks*, pages 19–28. ACM, 2010.

[3] Siegfried Benkner and Gerhard Engelbrecht. A generic qos infrastructure for grid web services. In *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 141–141. IEEE, 2006.

[4] Roland Bless and M Rohricht. Secure signaling in next generation networks with nsis. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[5] Cynthia Irvine and Timothy Levin. Toward a taxonomy and costing method for security services. In *Computer Security Applications Conference, 1999.(ACSAC'99) Proceedings. 15th Annual*, pages 183–188. IEEE, 1999.

[6] Ana Nieto and Javier Lopez. A context-based parametric relationship model (cprm) to measure the security and qos tradeoff in configurable environment. In *IEEE International Conference on Communications (ICC)*, pages 755–760. IEEE, 2014.

[7] Ana Nieto and Javier Lopez. Analysis and taxonomy of security/qos tradeoff solutions for the future internet. *Security and Communication Networks*, In Press.

[8] Tarik Taleb, Yassine Hadjadj Aoul, and Abderrahim Benslimane. Integrating security with qos in next generation networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.

[9] Rehana Yasmin, Eike Ritter, and Guilin Wang. An authentication framework for wireless sensor networks using identity-based signatures. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 882–889. IEEE, 2010.

[10] I-Ling Yen, Hui Ma, Farokh B Bastani, Hong Mei, et al. Qos-reconfigurable web services and compositions for high-assurance systems. 2008.