Modelado de Arquitecturas Interoperables Seguras

Juan Carlos Ramos[1], Marta Castellaro[1], Susana Romaniz[1]

[1]Universidad Tecnológica Nacional, Facultad Regional Santa Fe, 3000 Santa Fe, Argentina
{jcramos, mcastell, sromaniz}@frsf.utn.edu.ar
http://www.frsf.utn.edu.ar

Abstract. Web Service es una de las tecnologías y estándares comúnmente usada para implementar SOA. Pero, aplicar una capa de Web Service sobre aplicaciones legadas o componentes, no garantiza las propiedades SOA requeridas. Es necesario disponer de un enfoque sistemático y completo que tenga en cuenta los requerimientos de negocio y siga las prácticas recomendadas. La interoperabilidad es el principal problema a resolver a la hora de diseñar y desarrollar sistemas basados en servicios, pero esta facilidad de inter-operar no está completa si no se consideran los aspectos de seguridad en esta interoperación. En este trabajo se realiza primeramente un análisis y selección de propuestas para el modelado de arquitecturas basadas en servicios. Luego se analizan propuestas existentes para tratar el aspecto de seguridad en arquitecturas de software interoperables, y se discuten algunas propuestas para modelar específicamente la seguridad en arquitecturas de software. Del análisis se desprende que una manera adecuada para lograr modelar un esquema de interoperabilidad.

Keywords: Interoperabilidad, SOA, Seguridad, Modelado de Arquitecturas.

1 Modelado de Arquitecturas Basadas en Servicio

Web Service es una de las tecnologías y estándares comúnmente usada para implementar una Arquitectura Orientada a Servicios (*Service Oriented Architecture* - SOA). Pero, aplicar una capa de Web Service sobre las aplicaciones legadas o componentes no garantiza las propiedades SOA requeridas, tales como alineación al negocio, flexibilidad, acoplamiento débil, y reusabilidad. Por esto, es necesario disponer de un enfoque sistemático y completo que tenga en cuenta los requerimientos de negocio y siga las prácticas recomendadas.

Se trata de analizar cómo modelar las necesidades de interoperación entre sistemas y el cumplimiento de las mismas por las arquitecturas de software.

Existen diferentes propuestas para modelar SOA: basadas en distintas metodologías, con alcance a distintas fases del desarrollo de software, que emplean distintos procesos, usan distintos modelos, algunas públicas y otras no, como muestra la Tabla 1.

Pero para tener una solución de interoperabilidad exitosa es necesario considerar todas las capas (aplicación y tecnología). Así, se plantea que la interoperabilidad debe tenerse encuentra en las primeras fases del proceso de desarrollo de software, y en par-

Propuesta	Basada en	Alcance (Fases/Pasos)	Proceso	Modelos	Es Pública
IBM-SOAD [2]	OOAD, CBD, BPM	Conceptualización, Categorización, Agregación, Políticas y extracción de servicios	-	-	Sí
IBM-SOMA [3]	-	Identificación, especificación, realización, procesos de negocio, y componentes realizadores de servicios	-	-	Sí
SOA RQ	RUP	Principio, Elaboración, Construcción, Transición y Concepción.	Iterativo e Incremental	UML	No
CBDI-SAE [4]	-	Consume, provee, gestiona, y habilita servicios	-	-	No
SOAF [5]	-	Elicitación, identificación, definición, realización, hoja de ruta, y planificación de servicios	Top-down y Bottom-up	-	Sí
SOUP [6]	RUP, XP	Inicio, Definición, Diseño, Construcción, Distribución y Soporte de servicios	Iterativo e Incremental	-	Sí
SODDM [7]	RUP, DBD, y BPM	Fase principal, y fases complementarias.	Iterativo e Incremental	-	Sí
SOAD [8]	-	Análisis y Diseño	Secuencial	Funcionales	Sí
BPMN-PEL [9]	BPMN	Análisis y Diseño	Secuencial	UML	Sí
MSA [10]	-	Descubrimiento de servicios (Qué, Quién, Porqué, y Cómo)	Top-Down	-	Sí

Tabla 1.Características de las metodologías para modelar SOA.

ticular en las fases de análisis y diseño, para especificar las necesidades de interoperabilidad en las capas de arquitectura de negocio y arquitectura de aplicación.

En la sección 2 se analiza el tema de la seguridad en el entorno SOA. En la sección 3 se presentan propuestas para diseñar soluciones interoperables seguras utilizando los enfoques de SOA y Web Services. En la sección 4, se presentan propuestas para considerar la seguridad en el desarrollo de software independientemente si estos usan SOA o no. En la sección 5 se hace una conclusión de los modelos presentados, y se sugiere una línea de trabajo a seguir para modelar interoperabilidad y seguridad para un sistema de software.

2 Seguridad

SOA es un entorno caracterizado por: ***Virtualización de Servicios*** (interfaces que realizan funciones de negocio reconocidas); ***Reutilización de Servicios*** (las aplicaciones implementan funcionalidades de un catálogo de servicios de preexistente); ***Intermediación de Servicios*** (los servicios registran sus interfaces con agentes que facilitan la accesibilidad por otras aplicaciones). Ofrece un marco para el desarrollo reduciendo

costo y tiempo, a través de la reutilización de código, pero que se pierde en cuestiones relacionadas con la seguridad; por ende, asegurar un entorno SOA es un reto [11].

Algunas de las cuestiones que deben analizarse son las siguientes:

- ***Control de los datos***: En el caso extremo, una aplicación SOA llega a ser simplemente una cadena de llamadas a servicios de propósito general realizadas a través de interfaces de programación bien estructuradas. Los servicios a la vez pueden interactuar con otros servicios, requiriendo el pase de datos sensitivos. Estos servicios de propósito general entran en conflicto con los requerimientos de seguridad específicos de las aplicaciones. Por ejemplo, en una aplicación que maneja datos sensitivos, estos datos deben ser protegidos en un nivel apropiado a través de todas las redes y sistemas que atraviesan durante su procesamiento. Debe haber autenticación mutua de los participantes y se deben aplicar los niveles de autorización adecuados. Además, los registros de auditoría y *logging* deben ser parte de la infraestructura.

- ***Control del agregado de servicios***: Las buenas prácticas para mantener entornos de procesamiento seguro establecen que deben aplicarse procedimientos de control de cambio formales cuando se agregan nuevo software o sistemas. Lo opuesto es la norma para SOA. Las organizaciones son alentadas a desarrollar y distribuir nuevos servicios. No resulta claro de qué manera una aplicación sería capaz de reconocer un servicio engañoso.

- ***Manejo de la autenticación***: En la mayoría de las implementaciones SOA (SOAP o MQ Series) no se realiza autenticación por defecto. Aún si se implementa Web Service Security, debe determinarse qué significa autenticación en un medio débilmente acoplado como SOA.
  Existen al menos dos enfoques arquitectónicos a considerar para habilitar o no la autenticación:
  Preservar la No-Seguridad Básica de SOA: construir una pared alrededor del conjunto de servicios e infraestructura de soporte para que todos se traten entre sí de una manera confiable. En entornos SOA complejos, esto puede tomar la forma de implementar un segundo (o tercer) ESB (Enterprise ServiceBus) con cada uno de los componentes y servicios que debe aumentar su nivel de seguridad.
  Desconfianza mutua: Este modelo aplica a SOA principios de redes abiertas bien establecidos. Cada participante debe autenticarse en la infraestructura, y con cada uno de los otros servicios. Debe mantenerse (en la infraestructura o en cada componente) el estado de autorización de qué participantes pueden acceder a qué servicios o a qué recursos. Además, se deben soportar registros de auditoría y logging.

- ***No existe un concepto de 'de principio a fin' (End-to-End)***: En SOA se utiliza una infraestructura de software para crear un modelo de bus de procesamiento que ayuda a la conexión dinámica, mediación y control de los servicios y sus interacciones. ESB define un nuevo tipo de middleware de integración de aplicaciones que se crea para actuar como un *backbone* de integración ubicuo, a través del cual fluyen los servicios y componentes de aplicación.

El problema en SOA es que no provee seguridad de principio a fin. En ESB, cada una de los componentes en una cadena de procesamiento desconoce, en cierta manera, el procesamiento que ocurre en los otros componentes. Aún si se implementaran mecanismos de autenticación, no hay concepto de control 'de principio a fin' en un camino de procesamiento SOA.
Una primera aproximación a una solución, no muy elegante, es hacer que las componentes pasen a través de ciertos servicios que realicen los controles.

- ***Alcanzar requisitos con componentes externos (comerciales)***: Comúnmente, en el desarrollo de SOA se comienza seleccionando productos de infraestructura populares. Mientras que con este enfoque se maximiza la integración de las aplicaciones, se eliminan del proceso de selección los requerimientos de negocio y requerimientos de seguridad de información. Es difícil incorporarlos en estos entornos.

Estos desafíos presentan situaciones problemáticas sobre la seguridad a la hora de desarrollar aplicaciones SOA, que deben ser resueltos en el proceso de desarrollo.

3 Modelado de Arquitectura de software interoperable y seguridad

En este apartado se presentan algunas propuestas existentes para considerar la seguridad en el diseño de arquitecturas interoperables. Estas proponen modelos y arquitecturas para incorporar la seguridad como parte de SOA o utilizando Web Services, que son dos enfoques comunes para tratar la interoperabilidad.

3.1 Arquitectura de Seguridad Orientada a Servicios (SOS)

En [12], se establece que el problema de la seguridad en las aplicaciones de software modeladas como orientadas a servicios se plantea por el hecho que ya no son aplicables los modelos existentes para tratar la seguridad desde el punto de vista de las aplicaciones convencionales (definición de perímetros y seguridad centralizada), y por lo tanto, los atacantes explotan estas debilidades.

Como solución a este problema se propone ver a la seguridad como un servicio desacoplado y que funciona como una componente más. La arquitectura *Service Oriented Security* (SOS) no sólo se enfoca en el perímetro, sino que también provee de un *framework* para analizar y manejar el riesgo de los activos del sistema (servicios, datos e identidades).

Esta arquitectura provee un conjunto de puntos de vistas que le permiten al arquitecto de seguridad construir un diseño de sistema holístico, basado en un conjunto de vistas:

- ***Vista de Identidad***: se ocupa de la generación, comunicación, reconocimiento, negociación, y transformación de identidad.
- ***Vista de Servicio***: se ocupa de los servicios, sus métodos y partes componentes.
- ***Vista de Mensajes***: se ocupa de la carga de los mensajes de los servicios.
- ***Vista de Despliegue***: se ocupa de la defensa en profundidad del modelo de cinco capas: física, red, servidor, aplicación y datos.

- ***Vista del Ciclo de Vida de Caso de Uso de Transacción***: se ocupa del flujo de comportamiento clave y las relaciones en un sistema y sus actores desde una perspectiva 'de principio a fin'.

Cada una de las vistas individuales está compuesta de elementos específicos del dominio, restricciones, amenazas, riesgos, vulnerabilidades, y contramedidas. Además, cada vista incluye un conjunto de patrones y principios arquitectónicos claves. La arquitectura resultante toma en cuenta los intereses de cada uno de los dominios y provee soluciones completas basadas sobre la gestión del riesgo de activos digitales tales como los datos y las identidades.

3.2 Service Oriented Security Architecture (SOSA)

En [13], se propones *Service Oriented Security Architecture* (SOSA), la que está basada en la arquitectura ESB. SOSA está construida sobre la idea de que en vez de crear componentes de seguridad pesados que estén integrados en la aplicación o el middleware, es mejor construir seguridad en servicios modulares que sean independientes de la plataforma, fáciles de testear y documentar, y que tengan un alto grado de reusabilidad.

Las características que deben contemplarse específicamente para incorporar seguridad en software para entornos interoperables aplicando WebService son:

- ***Autenticación***. Al solicitante del servicio se le debe reclamar que provea sus credenciales antes de acceder al Web Service. Las especificaciones más importantes que tratan este aspecto son WS-Security y SAML (single sign-on).
- ***Autorización.*** Establecer condiciones claras bajo las cuales una entidad tiene permiso para acceder a cierto Web Service. La autorización está guiada por XACML.
- ***Confidencialidad.*** La información debe ser protegida en su paso a través de los diferentes servicios. XML-Encryption y WS-Security tratan este aspecto.
- ***Integridad.*** Los mensajes no deben ser alterados en su camino. La integridad está guiada por XML-Signature y WS-Security.
- ***No repudio***. El proveedor del servicio debe ser capaz de proveer que un solicitante use un cierto WebService (no repudio de solicitante), y el solicitante debe ser capaz de determinar que la información que ha originado proviene de cierto proveedor de servicios (no repudio de proveedor). XML Digital Signaturetrata este aspecto.
- ***Privacidad***. Tanto el solicitante del servicio como el proveedor del servicio deben ser capaces de definir políticas de privacidad. Este aspecto es tratado por WS-Policy y WS-SecurityPolicy.
- ***Auditoría***. Los accesos y comportamientos de los usuarios deben seguirse para poder asegurar que las obligaciones establecidas son respetadas. La auditoría se realiza mediante registros de auditoría, y puede ser activa o pasiva.
- ***Confianza.*** El solicitante del servicio y el proveedor del servicio deben ser capaces de determinar si ellos confían uno de otro. La confianza se trata en WS-Trust.
- ***Contabilidad y Cobro de Cargos***. No son aspectos primarios de la seguridad del sistema, pero están fuertemente acoplados con las otras funciones de seguridad anteriores. Por ejemplo, el cargo de un servicio requiere que el servicio conozca la identidad del solicitante.

Propuesta arquitectónica

La propuesta arquitectónica de SOSA consiste básicamente en considerar las funciones de seguridad realizadas como pequeños servicios modulares, denominados *servicios de seguridad*.

Se utilizan las técnicas de *Enterprise Application Integration* (EAI) para integrar los servicios de seguridad junto con los Web Services que son protegidos. Para la flexibilidad, se utiliza un modelo ESB, el que soporta orquestación de servicios y coreografía de servicios, los que normalmente son fáciles de configurar. Además, implementa ruteo de mensajes y patrones de mediación, los que permiten construir funcionalidad de una manera totalmente transparente.

Los servicios de seguridad son realizados como mediaciones de ESB y, son encadenados juntos por medio del ruteo de mensajes. La mediación y el ruteo de mensajes son suficientes para diseñar sistemas de seguridad escalables, extensibles y fácilmente configurables. En la Figura 1 puede verse una realización de este sistema.

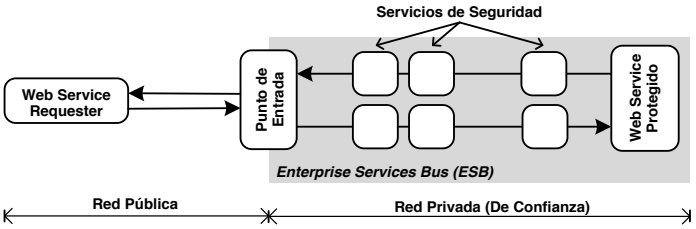

Fig. 1.Mensaje ruteador por ESB.

El modelo considera además que los servicios de seguridad confían unos de otros y que están ubicados en una red confiable (no implementan los servicios de autenticación entre sí, se asume que el entorno es seguro por definición).

La comunicación entre los servicios de seguridad debe contemplar además el pasaje del resultado del procesamiento intermedio que se da en la cadena de mensajes. Para ello, implementa un *patrón de anotación*: el primer servicio agrega el resultado del procesamiento intermedio al mensaje antes de despacharlo hacia el próximo servicio, y se lo denomina un mensaje anotado. Las anotaciones consistirán en, por ejemplo, atributos de identidad, decisiones de autorización, obligaciones, información de cuentas, etc.

Los principales servicios de seguridad posibles de implementar son:

- ***Autenticación***. Dos tipos de servicios: verificación e identificación. El primero verifica las credenciales (usuario, claves, etc.) del mensaje, mientras que el segundo es responsable de proveer atributos de identidad.

- ***Autorización.*** Tres tipos de servicios: *Policy Information Point* (PIP), *Policy Decision Point* (PDP), y *Policy Enforcement Point* (PEP). La tarea de un PIP es anotar los mensajes con atributos adicionales que el PDP puede requerir durante el proceso de decisión. La tarea de PDP es evaluar el mensaje, producir una decisión de autorización y anotar el mensaje con la decisión y, posiblemente, algunas obligaciones. La tarea del PEP es hacer cumplir las decisiones de los servicios PDP y cumplir con las obligaciones.

- ***Auditoría.*** Dos tipos de servicios: servicios que realizan auditoría pasiva, como un servicio de *logging*, y servicios que realizan auditoría activa, como un servicio de notificación.

- ***Servicios de criptografía***. El cifrado y la firma digital son tareas que requieren poder computacional significativo y por esto son ideales para distribuir en máquinas más potentes o con hardware especializado.

- ***Contabilidad.*** Si esto representa una tarea compleja, tiene sentido realizarla como un servicio independiente.

- ***Cobro.*** Si el cobro de servicio se realiza de manera inmediata, la tarea consiste en cobrar al solicitante según la información provista por servicio de contabilidad.

- ***Servicio de infraestructura***. Pueden ser necesarios otros servicios de mediación, especialmente si se piensa en acoplar diferentes servicios de seguridad: servicios de orquestación, servicios de transformación de mensajes, y servicios de almacenamiento de mensajes.

Dependiendo del contexto del sistema es posible implementar otros servicios de seguridad requeridos o utilizar otros niveles de granularidad de los mismos.

El siguiente paso es conectar los servicios de seguridad con la aplicación Web Service. Al estar implementados como servicios de mediación sobre un ESB, se pueden aplicar patrones de ruteo de mensajes para alcanzar la solución de seguridad.

Algunos patrones aplicables son [14]:

- ***Ruteo basado en contenido.*** Los mensajes se rutean entre los servicios en base a su contenido. Por ejemplo, los mensajes entrantes se rutean al servicio de identificación apropiado dependiendo del *token* de identificación que contienen.

- ***Ruteo de itinerario***. Se agrega al mensaje una ficha de ruteo que describe el itinerario; luego, el mensaje se reenvía de acuerdo a la ficha. Por ejemplo: autenticar - autorizar - auditar - WS protegido.

- ***Divisor / Sumador.*** El flujo del mensaje sigue diferentes caminos. Un requerimiento simple se puede dividir (*splitter*) y estos flujos paralelos resultantes se pueden sincronizar por medio de un sumador (*aggregator*), que combina los resultados.

En la Figura 2 puede verse un ejemplo de cómo la arquitectura combina todos los elementos. En este caso, luego de alcanzar el punto de entrada, a todos los mensajes

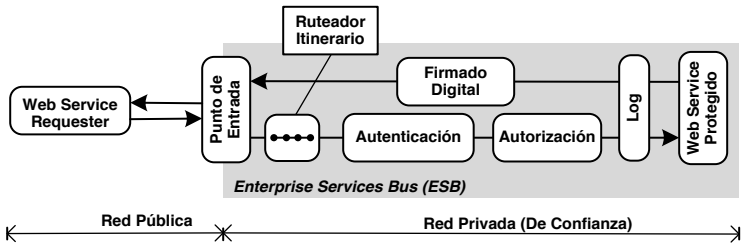

Fig. 2. Ejemplo de patrones de ruteo combinados.

les asigna un itinerario. De acuerdo a este itinerario los mensajes primero son autenticados, luego autorizados, luego registrados en el log, y recién en ese punto del proceso alcanzan el WebService Protegido. Cuando se devuelve un mensaje (respuesta), es

registrado en el log, firmado digitalmente y recién en ese punto del proceso es devuelto al solicitante.

3.3 Service Oriented Security ReferenceArchitecture (SOSRA)

Otra propuesta es *Service Oriented Security Reference Architecture* (SOSRA) [15], que está diseñada en base a la *Service Oriented Reference Architecture* (S3), utilizando un *Conceptual SOA Security Framework* [16] propuesto por los mismos autores.
Service Oriented Reference Architecture (S3 o SORA) es un modelo SOA de alto nivel presentado por IBM, el que muestra los bloques de construcción conceptual de una solución SOA, y las relaciones entre ellos. S3 puede utilizar como una base para especificar modelos de solución, y para modelar grandes sistemas SOA. El modelo puede observarse en la Figura 3.

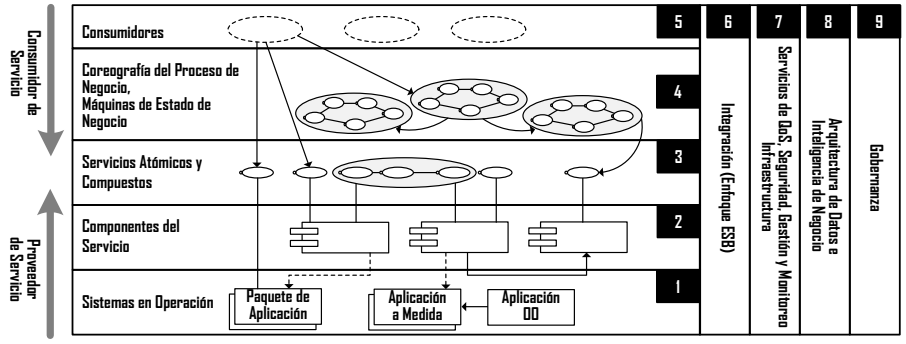

Fig. 3.*Service Oriented Reference Architecture*.

Cada capa tiene un aspecto lógico y físico. El aspecto lógico incluye todos los bloques de construcción arquitectónicos, decisiones de diseño, opciones, indicadores de performance claves, etc., mientras que el aspecto físico cubre la realización de cada aspecto lógico utilizando tecnología y productos. El modelo no posee una jerarquía estricta de capas, pero cada capa inferior brinda soporte a las capas superiores.

El *framework* SOA Security asiste para el modelado de todos los requerimientos de seguridad que pueden presentarse y puede ser aplicable a todas las capas de SORA. En la Figura 4 se muestra este *framework*.

Los autores proponen una nueva *Service Oriented Security Reference Architecture*(SOSRA) de acuerdo al *framework* de seguridad anterior (Figura 4) y los requerimientos de seguridad para el modelo S3 (SORA) (Figura 3), que se detalla a continuación:

Las primeras seis capas principales (consumidor, proceso de negocio, servicio, componente de servicio, operacional, y ESB) se ubican en la capa superior de la arquitectura, y luego se incorporan los servicios de seguridad relevantes.

El cliente es autenticado en la capa de consumidor de acuerdo a la identidad definida. La capa del cliente determina si el usuario está autorizado a acceder a la capa deservicio o a la capa de proceso de negocio. Si es así, el requerimiento y la identidad se propagan al proveedor del servicio

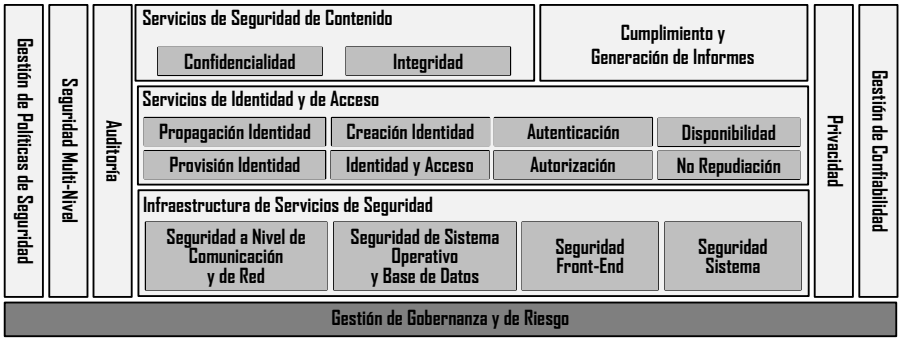

Fig. 4. ***Framework SOA Security*****.**

Todos los eventos deben registrarse como información de auditoría para verificar la performance de la capa de consumidor de acuerdo a las políticas. Esta información debe almacenarse en almacenamientos de auditoría y debe aplicarse a todas las capas.

Es necesario disponer de servicios confiables para establecer relaciones de confianza, a fin de evitar ingresos (sign in) repetidamente. Todos los mensajes durante la transformación deben protegerse y debe garantizarse la confidencialidad, la integridad y la privacidad de los mismos. Debe garantizarse la disponibilidad del servicio.

En las propuestas analizadas en 3.1 y 3.2, puede verse claramente cómo el enfoque para considerar la seguridad es tratar a ésta como un servicio más. Y de esta manera integrarse fácilmente con la propuesta base (Web Service o SOA) que trata al resto de las funciones del sistema como servicios, los que serán considerados a futuro.

En 3.3, puede verse un *framework* de nivel superior, donde la seguridad se trata también como un servicio, pero tiene una implementación en una arquitectura de capas, donde todas las operaciones/servicios deben pasar por la capa de seguridad obligatoriamente antes de poder concretarse. También este *framework* define explícitamente las demás capas que la arquitectura de software debe considerar para que sea una solución completa.

4 Modelado de Sistemas de Aplicación Seguros

Tal como se indicó al principio, entendemos que la seguridad debe considerarse en todo el proceso de desarrollo de software, independientemente de si se está utilizando el concepto de servicios o no (un enfoque convencional). Para esto presentamos las siguientes propuestas.

4.1 Modelado de la evolución de sistemas de aplicación seguros

La propuesta de [17] consiste en un enfoque sistemático para evolucionar de un sistema no seguro a un sistema seguro, al considerar la seguridad en todo el proceso de desarrollo de software.

Para que el sistema sea más fácil de mantener, los aspectos de seguridad en el modelo de requerimientos de software y arquitectura de software deben estar separados de los aspectos propios de la aplicación. Si bien los requerimientos de seguridad normalmente son considerados como 'requerimientos no funcionales', estos deben transformarse en 'requerimientos funcionales' para poder implementar los servicios de seguridad. Estos requerimientos deben ser modelados en forma separada.

Para la captura de requerimientos y el modelado de las soluciones se utiliza UML.

Los requerimientos de seguridad se capturan en casos de uso de seguridad y se los encapsula en *objetos de seguridad*. Los casos de uso de seguridad se extienden a partir de los casos de uso de la aplicación si se satisfacen las condiciones de requerimientos de seguridad. En forma similar, los objetos de seguridad se especializan a partir de los objetos de aplicación si el sistema necesita ser seguro.

La vista principal para modelar la arquitectura de software es la de *componentes* y *conectores* (C&C) [18], donde los componentes realizan los aspectos funcionales del sistema, y los conectores se encargan de los aspectos de comunicación entre ellos. Un conector actúa en representación de los componentes en términos de la comunicación entre componentes, encapsulando los detalles de la comunicación inter-componente. El rol de un conector es sincronizar mensajes de comunicación entre componentes. Para soportar la seguridad en el nivel arquitectónico, se extiende el rol de los conectores, agregándoles objetos de seguridad, *conectores seguros*, a medida que la arquitectura avanza hacia una arquitectura segura.

Los objetivos de seguridad de confidencialidad, integridad, control de acceso y no repudio, por ejemplo, pueden lograrse utilizando conectores seguros que implementen estos servicios. Si un servicio de confidencialidad prevé que información secreta está siendo observada por participantes no autorizados, se puede implementar un conector seguro que encapsule sistemas de cifrado. Un servicio de integridad se puede implementar utilizando *message digest* (MD) o *message authentication code* (MAC). Un servicio de control de acceso puede implementarse empleando *mandatory access control* (MAC). Finalmente, un servicio de seguridad para no repudio puede realizarse utilizando firma digital.

La propuesta se orienta a que las componentes de la arquitectura de software tengan mínimos cambios cuándo estos evolucionan hacia una arquitectura de software segura.Algunos servicios de seguridad requerirán de ciertos datos para poder realizar su función de seguridad. En estos casos, se procurará que el componente que se conecte a un conector seguro provea de los datos requeridos por el servicio seguro.

En la Figura 6 se muestre una Arquitectura de Software Segura con sus componentes y conectores en su representación como objetos concretos.

Fig. 5. Componentes y conectores en una arquitectura de software seguro.

Para modelar los aspectos estáticos y dinámicos del sistema, y su evolución hacia un sistema seguro, se aplican los siguientes criterios.

Modelado estático

El modelado estático en UML se realiza a través de diagramas de clases. Utilizando la jerarquía de generalización/especialización, es posible hacer evolucionar las clases de la aplicación hacia clases que contienen los servicios de seguridad requeridos.

Modelado dinámico

A través de un diagrama de colaboración se muestran los objetos que participan en cada caso de uso, y la secuencia de mensajes entre ellos. La evolución de seguridad de una aplicación se representa en un modelo de colaboración utilizando una secuencia de mensajes alternativos, los que conforman un camino condicional consistente de objetos y mensajes. Un caso de uso de seguridad extiende un caso de uso normal cuando se cumplen las condiciones de seguridad. Esto hace que se deba modificar el correspondiente diagrama de colaboración a través de una secuencia de mensajes alternativos. La condición de requerimiento de seguridad garantiza la ejecución de la secuencia alternativa.

Los requerimientos de seguridad capturados en casos de uso separados se encapsulan en objetos de seguridad separados. Cuando la aplicación necesita ser segura, los objetos no seguros se comunican con los objetos de seguridad a través de secuencias de mensajes alternativos.

En la Figura 7 [17] se observa un diagrama de colaboración para un Browser de Catálogo con Seguridad de Control de Acceso, que es la extensión de seguridad del caso de uso normal.

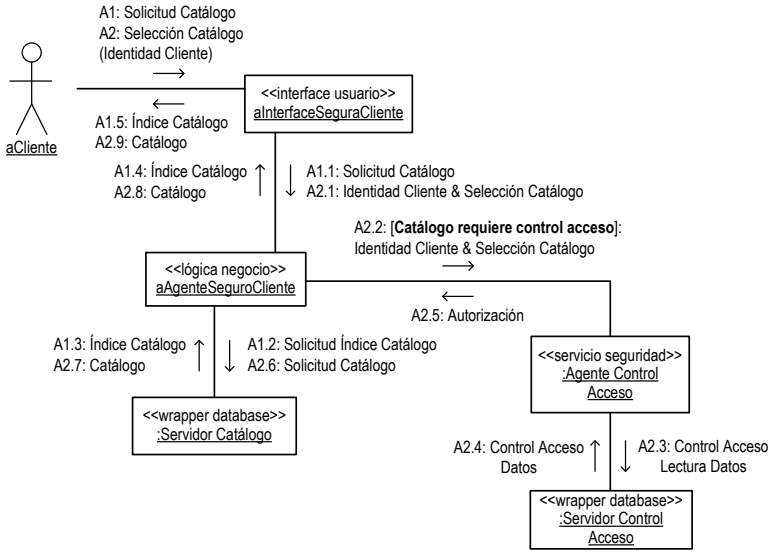

Fig. 6. Ejemplo de un diagrama de colaboración.

Las vistas anteriores pueden considerarse las vistas primarias, cuya combinación permite modelar la evolución de una arquitectura de software no segura a una arquitectura de software segura, a medida que se van presentando las necesidades de segu-

ridad. Se puede aplicar la propuesta para generar directamente una arquitectura segura que tiene separados los conceptos bases del negocio de los de seguridad.

4.2 Un enfoque basado en vistas para especificación de SOSA

Otra propuesta considerada es que se presenta en [19], donde se propone una especificación de un modelo de SOSA incorporando diferentes vistas interrelacionadas: a) vista de ingeniería de seguridad; b) la vista de servicio de seguridad; y c) la vista de integración.

Vista de Ingeniería de Seguridad

Aquí, el foco es el análisis y descubrimiento de servicios así como también el diseño de los servicios. La fase de implementación y distribución sólo es necesaria para componentes adaptadores especializados en mapear interfaces especializadas.

El objetivo del análisis de seguridad es especificar los requerimientos de seguridad de la aplicación. Una arquitectura de seguridad existente implementa requerimientos de seguridad de aplicaciones pre-existentes, orientadas a servicios. Por esto, la información sobre los requerimientos de seguridad, las amenazas identificadas y sus correspondientes ataques, así como las implementaciones para mitigar o prevenir los ataques, necesita ser documentada y catalogada. De esta manera, los desarrollos tienden a brindar soporte para el análisis de requerimientos de seguridad de nuevas aplicaciones.

Durante la fase de diseño, los requerimientos de seguridad se mapean con un diseño de seguridad apropiado. El diseño de seguridad consiste en las estructuras estáticas de software (componentes y servicios de seguridad) y sus interacciones dinámicas, las que se agregan al diseño funcional de la aplicación. El empleo de patrones de seguridad permite aplicar mejores soluciones prácticas a problemas de seguridad recurrentes [20]. Una arquitectura de seguridad debe abstraer las medidas de seguridad implementadas, especificarlas mediante el uso patrones de seguridad y relacionarlas con los requerimientos de seguridad previos, mientras mantiene la implementación oculta al consumidor del servicio de seguridad. En vez de agregar directamente un patrón de seguridad al diseño de la aplicación, el patrón debe especificarse en forma separada, como parte de la arquitectura de seguridad. Se puede utilizar anotaciones semánticas para reflejar la ubicación en el diseño de la aplicación, y cuáles medidas de seguridad van a agregarse. Esto separa el diseño de seguridad del diseño de la aplicación, de acuerdo al paradigma de separación de intereses, lo que además permite que la especificación de los patrones de seguridad se desarrolle en forma independiente del diseño de la aplicación, y sirve para un mejor re-uso del diseño de seguridad en diferentes diseños de aplicaciones.

Las políticas de seguridad son una parte importante en una arquitectura de seguridad. Los requerimientos de seguridad de una aplicación determinan la clase de políticas de seguridad que la arquitectura de seguridad debe soportar. El reuso de modelos de políticas existentes de una arquitectura de seguridad ayuda a los desarrolladores a elegir el modelo adecuado a su caso. Algunas aplicaciones pueden usar RBAC [21], mientras otras pueden requerir una política de control de acceso de granularidad más fina utilizando control de acceso basado en atributos [22].

Vista de Servicios de Seguridad

Esta vista pone foco en proveer los servicios de seguridad de una arquitectura de seguridad. Actúa como un mediador entre la vista abstracta de ingeniería de seguridad y la vista de nivel inferior de integración. Los patrones de seguridad abstractos se mapean con un conjunto específico de estándares y tecnologías de seguridad. Si bien la arquitectura de seguridad tiende a centralizar la mayoría de las funcionalidades relacionadas con la seguridad, hay ciertos componentes orientados a la seguridad que son difíciles de separar de las aplicaciones funcionales.

Un ejemplo típico son los *Policy Enforcement Point* (PEP), los que realizan el resultado de un *Policy Decision Point* (PDP) [23]. La vista de servicios necesita especificar qué grado de integración de componentes relacionados a la seguridad y componentes y servicios de la aplicación se realiza. Esto se hace normalmente utilizando protocolos de interacción específicos, los que pueden ser modelados utilizando diagramas de secuencia de UML.

Para interactuar con terceros se requiere interoperabilidad entre los servicios de seguridad para el intercambio de información de seguridad. Usar estándares basados en XML (tales como SAML, XACML o WS-Security) son buenas elecciones para la interoperabilidad con socios externos, pero pueden tener efectos negativos sobre la performance cuando se los utiliza internamente. Por este motivo, la vista de servicios de seguridad necesita proveer especificaciones de los servicios de seguridad disponibles, el uso de estándares relacionados así como también estándares o tecnologías propietarias, tanto para uso interno como externo.

El uso de estándares abiertos requiere de una adecuada documentación, ya que estos proveen un alto grado de flexibilidad.

Vista de Integración de Seguridad

En esta vista se muestra cómo se reestructuran productos de seguridad propietarios existentes a servicios de seguridad, mediante la centralización de componentes de seguridad de aplicaciones existentes [19].

Las otras vistas se construyen sobre productos de seguridad existentes, y como su funcionalidad sólo puede ser accedida como servicios es necesario abstraerlas de estos para poder utilizarlas en las otras vistas. Dado que esto es de naturaleza técnica, esta vista está orientada expertos de seguridad de la organización [24]. El modelo propuesto puede verse en la Figura 8.

La propuesta presentada en 4.1 se orienta a tratar los aspectos de seguridad como una extensión de los conceptos base de la aplicación (funciones y clases), en tanto que la presentada en 4.2 utiliza un enfoque a vistas para separar diferentes intereses de la aplicación. Fundamentalmente, hace una propuesta fuerte en utilizar patrones de seguridad dar solución a los problemas de seguridad de la aplicación (el área de patrones de seguridad es un área de investigación de los autores de este trabajo). Ambas propuestas utilizan UML como un lenguaje para modelar los sistemas en todos sus aspectos.

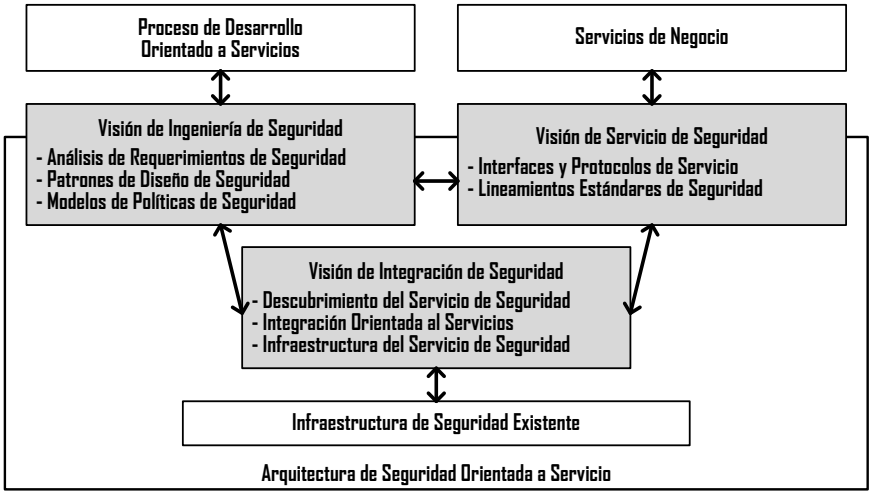

Fig. 7. Vista de integración de servicios de seguridad.

5 Conclusiones

En este trabajo se hace una presentación, principalmente, de propuestas para considerar la seguridad en el diseño de arquitecturas interoperables (aparado 3), como así también para considerar la seguridad en el diseño de software general (apartado 4). De la revisión de las propuestas de arquitecturas y formas de modelar arquitecturas interoperables seguras, se desprende que una manera adecuada para lograr modelar un esquema de interoperabilidad es tratar a todos los elementos necesarios como servicios. Debido a que la seguridad es uno de estos elementos, ésta debe modelarse también como un servicio, lo que permite su fácil incorporación al diseño del sistema. El uso de UML como lenguaje de modelado también es un aspecto relevante que facilita la comunicación.

De acuerdo con esto, tomamos como referencia los trabajos de [12] y [13], por lo cual el modelado de la arquitectura deberá realizarse tratando a todos sus elementos como servicios. Para la documentación de la arquitectura se propone utilizar UML como un lenguaje de uso común, y en particular Vistas de Componentes y Conectores de acuerdo a la propuesta de [17] que permiten visualizar la interconexión entre los componentes y las implementaciones de seguridad requerida, y generar las vistas necesarias como lo propone [19].

Actualmente, los autores estamos enfocados en la definición del proceso sugerido a seguir para modelar y documentar arquitecturas de software interoperables conforme las pautas aquí presentadas, lo que se constituirá en resultado de un próximo trabajo.

Referencias

1. Ramollari E., Dranidis D. and Simons A.: A survey of service oriented development methodologies.The 2nd European Young Researchers Workshop on Service Oriented Computing.UK.2007. Pag. 75 a 80.

2. Gee C.: Elements of service oriented analysis and design. IBM developers Works. 2004.http://www.ibm.com/developerworks/library/ws-soad1/(acceso 06/05/2015)
3. Arsanjani A.: Service-oriented modeling and architecture how to identify, specify, and realize services for your soa. IBM developers Works. 2004.
4. Sprott D.: SAE2 framework for application modernization. CBDI Journal. 2009.
5. Thang Le Dinh L.: Towards an approach for modeling interoperability of information systems. IEEE Xplore Digital Library- MINCYT. Research, Innovation and Vision for the Future, 2007 IEEE International Conference on. 2007. Pag. 22 a 28.
6. Mittal K.: Service oriented unified process (soup). 2010.
7. Papazoglou P.: Service-oriented design and development methodology. Int. J. of Web Engineering and Technology (IJWET). 2006.
8. Erl T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR. August 2005.
9. Abeck S.: Development of soa-based software systems an evolutionary programming approaches. Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006).2006.
10. Jones S.: A methodology for service architectures. 2005.
11. Gossels J., Mackey R.: Service oriented architecture: Security challenges. SystemEXPERTS Corporation.2005.
12. Peterson G.: Service oriented secure architecture. Information Security Bulletin. 2005. Vol 10. Pag. 235.
13. Opincaru C, Gabriela G.: A service-oriented security architecture. Enterprise Modelling and Information Systems Architectures - Concepts and Applications. Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'07). Germany, October 8-9, 2007
14. Hohpe G., Wolf B: Enterprise Integration Patterns. Designing, Building, and Deploying Messaging Solutions. Pearson Education-Addison Wesley.2004. ISBN: 0-321-20068-3
15. Ibrahim S.: A service oriented security reference architecture. International Journal of Advanced Computer Science and Information Technology , Vol. 1, No.1, pp. 2531. 2012.
16. Khezrian M.: Enabling security requirements for enterprise service-oriented architecture. Int. J. on Recent Trends in Engineering and Technology, vol. Vol. 6. Nov 2011.
17. Shin M., Gomaa H.: Software Modeling of Evolution to a Secure Application: From Requirements Model to Software Architecture. Science of Computer Programming, Volume 66, Issue 1, April 2007, pp. 60-70.
18. Clements P.: Documenting Software Architectures. Addison Wesley. 2011.
19. Dikanski A., Abeck S.: A view-based approach for service-oriented security architecture specification. The Sixth International Conference on Internet and Web Applications and Services. ICIW 2011. St. Maarten, The Netherlands Antilles.
20. Yoshioka N., Washizaki, H. and Maruyama, K. :A survey on security patterns, Progress in Informatics, No. 5, pp35-47, 2008
21. Sandhu R. S.: Role-based Access Control. Advances in Computers. VOL. 46. IEEE. 1996.
22. Yuan E.: Attributed Based Access Control (ABAC) for Web Services. Proceedings of the IEEE International Conference on Web Services (ICWS'05).2005.
23. Schumacher M., y otros: Security Patterns Integrating Security and Systems Engineering. John Wiley & Sons Ltd. 2006.
24. Dikanski A.: Integration of a security product in service-oriented architecture. Proc. third Intl. Conf. Emerging Security Information, Systems and Technologies (SECURWARE 09).2009.