Aportes para una Internet de las Cosas Seguras

Miguel Solinas
Departamento de Computación
Facultad de Ciencias Exactas Físicas y Naturales - UNC
Av. Vélez Sarsfield 1611
miguel.solinas@unc.edu.ar

Javier Alejandro Jorge
Desarrollo en electrónica e informática
Instituto Nacional de Tecnología Industrial
Av. Vélez Sarsfield 1561, Córdoba
jjorge@inti.gob.ar

Eduardo Casanovas
Departamento Computación e Informática
Facultad de Ingeniería - IUA
Av. Fuerza Aérea 6500 - Córdoba
ecasanocas@iua.edu.ar

Carlos Tapia
Departamento Computación e Informática
Facultad de Ingeniería - IUA
Av. Fuerza Aérea 6500 - Córdoba
ctapia@iua.edu.ar

Resumen

Este trabajo surge de un proyecto de investigación que cuestiona el impacto de la seguridad de los sistemas embebidos sobre las infraestructuras críticas. Una encuesta reciente con un diagnóstico preocupante sobre la seguridad de los sistemas embebidos que se están construyendo es el disparador para precipitar esta publicación. Todos los fabricantes aspiran a conectar sus desarrollos a Internet y sabemos que Internet es un espacio esencialmente inseguro. Luego se presentan tres propuestas esenciales para mejorar la seguridad en la construcción de sistemas compuestos por hardware y software que censan, procesan y actúan sobre el mundo físico.

1. Sistemas ciber físicos

El uso más visible de las computadoras (hardware y software) es el procesamiento de la información. Las utilizamos para escribir y editar documentos de todo tipo, buscar información en la web, comunicarnos por correo electrónico y consultar datos financieros. Pero la mayoría de las computadoras en funcionamiento son menos visibles.

La mayoría de las computadoras funcionan embebidas en automotores como sistemas de control de motores, frenos, cinturones de seguridad, airbag y audio. Codifican digitalmente la voz y construyen una señal de radio para enviarla desde un teléfono celular a una estación base y desde ella a otro teléfono celular. Controlan el horno de microondas, el refrigerador y el lavavajillas. Funcionan en impresoras domésticas e industriales. Comandan robots de líneas de montaje, controlan la generación de plantas de energía, los procesos en las plantas químicas y los semáforos de una ciudad. Buscan microbios en muestras biológicas, construyen imágenes del interior del cuerpo humano y miden sus signos vitales. Procesan señales de radio del espacio buscando supernovas. También controlan autos, aviones, drones y trenes.

Todas estas computadoras, “menos visibles”, se denominan sistemas embebidos y el software que ejecutan se conoce como software embebido.

Si bien los sistemas embebidos existen desde la década del ’70, el tratamiento que se les dio para su diseño y construcción fue el tratamiento habitual para una computadora utilizada para procesar información.

Ya en este siglo XXI, gracias a la evolución y madurez de los sistemas embebidos, se superaron la mayoría de sus problemas de limitación de recursos. Y el principal problema en su diseño ha pasado a ser la interacción con el mundo físico.

A partir de esa nueva problemática surge el término Cyber Physical Systems (CPS) como el concepto que contiene a los sistemas embebidos y enfoca los principales desafíos de diseño e implementación sobre la interacción con el mundo físico.

Estos nuevos sistemas complejos, son la base para la construcción, entre otras cosas, de las anunciadas ciudades inteligentes. Millones de sensores/actuadores distribuidos en dispositivos que prometen ayudar a mejorar la experiencia de vivir en las grandes ciudades.

También son la base para el desarrollo de la Internet de las Cosas o Internet of Things (IoT), la Industria 4.0, la Internet Industrial y futuras innovaciones que pudieran surgir.

El desarrollo de estos conceptos y la propuesta de nuevas aplicaciones han sido tan vertiginosos que hoy se habla del concepto de "neblina", como una especie de "nube" pero más próxima a la superficie terrestre. Viviremos sin darnos cuenta y de forma permanente inmersos en una "neblina" de sistemas embebidos.

2. Internet de los sistemas críticos

La visión que tendrá el habitante de las grandes ciudades, de una experiencia con un entorno "inteligente", donde los servicios que antes funcionaban desconectados de Internet ahora de disputan su presencia en la red, trae aparejado el riesgo de sumar elementos que controlan nuestra vida cotidiana a un mundo **esencial y profundamente inseguro**. Si hay una propiedad que podemos observar de Internet, es su inseguridad. Es suficiente un buscador web y algunas pocas palabras claves para que incluso un neófito encuentre evidencias de esta afirmación.

Ese mundo inseguro se construyó a partir de tres premisas básicas:

- La primera es la excesiva importancia al **"time to market"**, lo esencial es llegar primero para capturar la mayor porción del mercado.
- Lo segundo fue producto directo del primero, las empresas trabajaron desde 1969 (nacimiento de la Internet) para satisfacer la necesidad de entregas **(release o muerte)** no de "entregas seguras". Y sus colaboradores cumplieron, llegaron primero, pero nunca fue una motivación llegar con un producto seguro.
- Ese desinterés por la seguridad generó el tercer principio, **un vació de herramientas** y métodos para diseñar, implementar, validar y mantener sistemas seguros.

Esto tres factores son determinantes de la inseguridad de la Internet que nos rodea y tiene garantizada su continuidad.

Ahora bien, nos preguntamos ¿qué son las infraestructuras críticas? y ¿qué tienen que ver con la "neblina"?

Las infraestructuras críticas son aquellas que brindan los servicios esenciales para una sociedad u organización moderna. Ponerlas en riesgo, es poner el riesgo la gobernabilidad de una nación, de una organización o su estilo de vida. Por ejemplo, el "*Presidential Policy Directive-21*" (PPD-21) [1], referida a la Seguridad y Resiliencia de Infraestructuras Críticas de USA, identifica 16 sectores:

1. Industrias químicas,
2. Infraestructuras comerciales,
3. Comunicaciones,
4. Fábricas,
5. Represas,
6. Industria base de defensa,
7. Servicios de emergencia,
8. Energía,
9. Servicios financieros,
10. Alimentación y agricultura,
11. Instalaciones de gobierno,
12. Salud pública y privada,
13. Tecnologías de la Información,
14. Materiales y reactores nucleares,
15. Sistemas de transporte,
16. Agua y saneamiento.

Los CPS (o la más familiar IoT) si bien van a mejorar muchos aspectos de nuestras vidas, también harán críticas muchas actividades que hasta hoy no lo son. Por lo que este listado se ampliará cuando el mundo sea más "inteligente" gracias a las contribuciones de los Cyber Physical Systems de manera directa y a los sistemas embebidos de manera indirecta.

3. Seguridad de los sistemas embebidos

Nos referiremos a un informe reciente de Barr Group: 2017 Embedded Systems Safety & Security Survey. Barr Group es una Empresa de consultoría enfocada específicamente en la calidad, seguridad y la confiabilidad de sistemas embebidos.

Este año 2017 realizó la tercera encuesta anual global de mercado, con el objetivo de profundizar en el conocimiento sobre tendencias y prácticas que hacen a la confiabilidad y seguridad propiamente dicha en la industria de Sistemas Embebidos y elaborar un diagnóstico que permita realizar propuestas de mejora.

Se completaron 2.022 encuestas donde aproximadamente la mitad de ellas procedían de América del Norte, una cuarta parte de Europa, un 14% de Asia y un 9% para el resto del mundo.

En primer lugar, fueron eliminados 147 encuestados si la persona encuestada no tenía experiencia directa con el diseño, o si los encuestados estaban en la universidad. A ellos se sumaron 80 encuestados más que no estaban directamente involucrados con el diseño de Sistemas Embebidos. Finalmente, hubo 69 encuestados, que presentaron respuestas inconsistentes en los detalles sobre el proyecto actual, por lo que se asumió que no estaban realmente comprometidos con el diseño. Así la encuesta quedó conformada por 1.726 profesionales ingenieros calificados, fuertemente involucrados con los procesos de diseño de Sistemas Embebidos. Se pueden consultar mayores detalles sobre la encuesta en [2].

Las principales conclusiones a la que llega la encuesta son las siguientes:

- La industria está construyendo Sistemas Embebidos que forman parte de sistemas que frente a una falla pueden poner en riesgo la salud y la vida de las personas sin tener en cuenta aspectos fundamentales para minimizar el riesgo de que el peor caso ocurra. Y se refiere a sectores de la industria regulados como la industria automotriz, equipos médicos y electrónica de consumo.
- El 60% de los dispositivos que hoy construye la industria, en algún momento de su vida útil se conectará a Internet. Eso es un riesgo enorme conociendo la problemática de seguridad que padece la Internet.
- A pesar de que hay regulaciones y recomendaciones para mejorar la seguridad de los sistemas que se construyen, hay un tercio de proyectos que no las tienen en cuenta a pesar de que están construyendo sistemas críticos que pueden poner en riesgo la seguridad de las personas que los utilicen o las que se encuentren en su proximidad.
- Si bien la seguridad es una preocupación en todos estos proyectos, no se la aborda ni trata como un aspecto que debe ser considerado a la altura de un requerimiento funcional. Este criterio incluye a dispositivos críticos.

Se pueden obtener más conclusiones del informe pero creemos que estas cuatro son suficientes para abordar por un lado la propuesta de este trabajo y por otro la necesidad de reflexionar sobre qué haremos frente a una próxima Internet de las cosas inseguras.

4. Aspectos esenciales para la mejora de la seguridad en CPS

Cuando hablamos de la esencia de algo, el diccionario de la Real Academia Española, nos menciona en sus primeras acepciones “aquello que constituye la naturaleza de las cosas, lo permanente e invariable de ellas”, luego se refiere también en términos de “lo más importante y característico de una cosa”. Si pretendemos construir sistemas seguros, existen cuestiones que se transforman en esenciales.

En esta sección proponemos tres aspectos esenciales para la construcción de CPS seguros. No excluyen otros criterios y recomendaciones que puedan realizarse y en todo caso siempre las complementarán.

Primero, la seguridad es un aspecto que debe ser considerado como un requerimiento funcional a nivel de sistema que luego se trasladará sobre sus componentes de software y hardware.

Segundo, se debe abordar la construcción de éste tipo de sistemas a partir del diseño y evaluación de modelos a nivel de sistema que luego impactarán sobre modelos de los diferentes componentes de software, hardware e interacción con el mundo físico.

Tercero, en la medida de lo posible, se deben utilizar las mejores prácticas en la construcción de la seguridad y con ello nos referimos a soluciones a problemas conocidos, concretamente patrones de diseño.

A continuación desarrollamos estas tres ideas que son el núcleo de nuestro trabajo.

4.1. La seguridad como requerimiento funcional

Hace más de diez años que diversos autores [3] [4] [5] [6] proponen diferentes estrategias para abordar el tratamiento de la seguridad en términos de requerimientos equivalentes a los funcionales. Me referiré brevemente a la propuesta de [6] y a la utilización de herramientas de aplicación común para un sistema como para un componente de software.

Fernández describe una metodología de construcción de software seguro en donde la idea fundamental es aplicar principios de seguridad a cada etapa del desarrollo de software, y chequear en cada etapa el cumplimiento de esos principios, en coincidencia con la propuesta de [3] que destaca el test en cada etapa de un desarrollo orientado a objetos. En la Figura 1 se puede ver un ciclo de vida de desarrollo de software seguro, donde las flechas blancas indican dónde debe ser aplicada la seguridad y las flechas negras indican dónde debe chequearse el cumplimiento de los principios de seguridad.

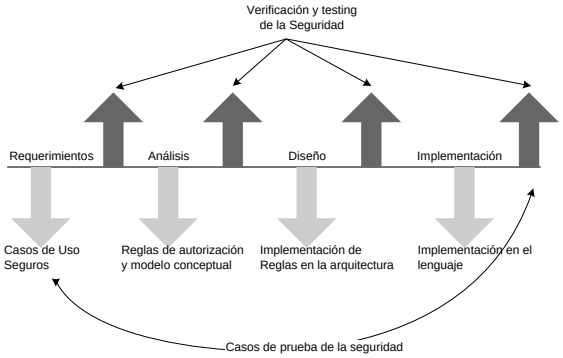

Figura 1. – Ciclo de vida de un desarrollo de software seguro.

De la etapa de requerimientos se obtienen casos de uso. De la etapa de Análisis se obtienen reglas de autorización que se aplican al modelo conceptual. En la etapa de Diseño se aplican las reglas sobre la arquitectura. En la implementación, Despliegue y Mantenimiento se requieren lenguajes que permitan implementar las contramedidas identificadas. Se debe

destacar que la verificación de la seguridad ocurre en cada etapa del desarrollo. En la Tabla 1 se muestran los métodos utilizados en cada etapa.

En particular, la adaptación de los Casos de Uso para el modelado de requerimientos a nivel de sistema y de componentes de software, ha dado lugar a un conjunto muy prometedor de aproximaciones en el área de modelado de requerimientos que atacan el problema de la seguridad. Para distinguirlos del Caso de Uso estándar, les han asignado varios nombres: casos de abuso, casos de mal uso, casos de uso hostiles, y casos de fiabilidad; estos últimos enfocados en las excepciones. Como elemento en común, intentan ver el sistema/software desde el punto de vista de un atacante. A los fines de este trabajo y utilizando el elemento en común, se los identifica a todos ellos como Casos de Mal Uso (CMU) o "Misuses Cases", si bien presentan diferencias.

Etapa	Método para implementar la seguridad
Requeri-mientos	Casos de uso basados en la identificación de roles y análisis de ataques
Análisis	Modelo de patrones de la semántica de análisis
Diseño	Aplicación coordinada de patrones en las múltiples capas de la arquitectura
Implementación	Incorporar aplicaciones COTS seguras

Tabla 1. – Resumen de los métodos utilizados en cada etapa.

Del mismo modo en que los Casos de Uso se utilizan exitosamente para elicitar requerimientos, los CMU se utilizan para identificar potenciales amenazas. A partir de ellas es posible elicitar requerimientos de seguridad o Casos de Uso de seguridad. La interacción del usuario autorizado con el sistema se representa simultáneamente con las interacciones del atacante. Y así como en los Casos de Uso las conexiones entre actor y acción se etiquetan con términos de "extiende" e "incluye", las conexiones en un CMU son etiquetadas con "amenaza" y "mitiga". De este modo, los CMU son el punto de partida para construir un conjunto de Casos de Uso seguros para contrarrestar cada una de las amenazas. Al igual que un Caso de Uso, cada CMU conduce un requerimiento y el correspondiente escenario de prueba para el software. De este modo gran parte del trabajo, invertido en construir los CMU, conduce al desarrollo de requerimientos funcionales de seguridad.

En la construcción de los CMU hay una analogía muy productiva con el siguiente juego: el mejor movimiento de las blancas consisten en pensar por anticipado la mejor movida de las negras. Por ejemplo, si el Caso de Uso que se desea analizar es el del robo de un automóvil, como se muestra en la Figura 2, el jugador de blancas es el propietario legal del vehículo y el jugador de negras el ladrón. La libertad del Conductor estará puesta en riesgo por el Ladrón, de modo que el Conductor necesita "Bloquear el auto". Este es un requerimiento derivado que mitiga la amenaza y ocurre en el nivel más alto de análisis. En el siguiente nivel de análisis se comienza con la respuesta del Ladrón frente a la acción del Conductor de "Bloquear el auto". Si el Ladrón logra acceder al auto y cortocircuitar el arranque del vehículo, está burlando el bloqueo del auto y por ende poniendo bajo amenaza el requerimiento derivado "Bloquear el auto". Para mitigar esta amenaza, se puede tomar la decisión de bloquear el arranque, solicitando el ingreso de una clave para desbloquearlo. Este es un nuevo **requerimiento funcional derivado**. Se puede observar un movimiento en zig-zag, balanceado entre las decisiones de jugar y contrajugar, propuesto por los usuarios (blancas) y el atacante (negras).

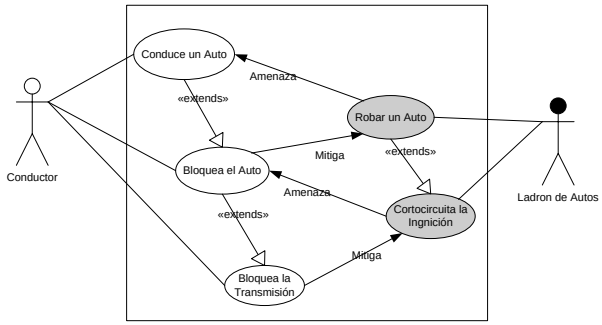

Figura 2. – Casos de Uso para "Seguridad de Automovil"

Los "requerimientos" surgidos a partir de considerar las decisiones del atacante, disparan extensiones de requerimientos funcionales, que se pueden mapear sobre Patrones de Seguridad (ver 4.3). Esta es una metodología perfectamente compatible con el diseño a nivel de sistema y de componentes de software. También es una metodología que se complementa con otras [7] [8]. Puede ser útil un juego de blancas y negras a fin de descubrir en una primera instancia un conjunto de potenciales extensiones que luego se mapean sobre Patrones de Seguridad, a partir de estudiar las decisiones de un atacante.

La obra [9] es una muy buena introducción al tema de CMU. El artículo hace una distinción entre CMU y casos de abuso, observando la "no intencionalidad" en la ocurrencia de los primeros y la "intencionalidad" en los segundos, y por ello mismo considerados hostiles. Es una distinción similar a la existente entre los riesgos y amenazas, pero no es una distinción estándar. También sugiere utilizar patrones de ataque a fin de ayudar a identificar CMU. Esta tarea debe ser realizada en equipo, donde participen ingenieros de software expertos en el dominio y expertos en el dominio de la seguridad.

Para una descripción concisa de la técnica, se puede consultar [10]. De esta obra es el siguiente párrafo sobre la descripción para contruir CMU:

"...para CMU, hacer una tormenta de ideas para identificar el modo en que los agentes negativos intentarán impedir o frustrar algunos de los pasos en la descripción del caso de uso; esto conduce a los principales CMU. Durante la sesión, los participantes deben enfocarse en identificar las múltiples formas en que un atacante podría causar daño en el caso de uso bajo estudio; luego podrán completarse otros detalles del ataque. Cada uno de estos modos de ataque es un candidato firme a un CMU.

El objetivo es identificar amenazas a la seguridad en cada una de las funciones, áreas, procesos, datos y transacciones involucradas en el caso de uso a partir de potenciales riesgos, como pueden ser los accesos no autorizados desde adentro y desde afuera; ataques de DoS; violaciones a la privacidad, confidencialidad e integridad y ataques malintencionados por parte de hackers. Además de estudiar los modos de ataque, el proceso podría también intentar descubrir posibles errores de los usuarios y la correspondiente respuesta de la aplicación. Frecuentemente estos errores podrían causar errores serios en la funcionalidad o en la seguridad de la aplicación. Identificando todas las acciones inapropiadas que podrían ocurrir, capturamos todas las acciones de uso anormal de la aplicación por parte de usuarios genuinos, en términos de accidentes o errores por descuido, o por parte de atacantes intentando quebrar o dañar el correcto funcionamiento de la aplicación."

4.2. Construcción de modelos para sistemas embebidos

La práctica de construcción de modelos se ha utilizado ampliamente a lo largo de la historia de la humanidad. Los modelos tratan de explicar el comportamiento de sistemas complejos mediante representaciones que los abstraen. Como ejemplo podemos citar diferentes modelos diseñados para explicar los movimientos celestes: el sistema de epiciclos, el sistema Ptolomaico, el sistema de Tycho Brahe, el sistema Compernicano, etc.

El objetivo de todos ellos fue predecir la evolución de la posición de los planetas. Considerados en conjunto representan un proceso de refinamiento del conocimiento de un dominio, donde un nuevo sistema reemplaza al más viejo cuando surgen evidencias, mostrando que el viejo sistema no se corresponde con la realidad.

La construcción de modelos ha sido, de este modo, una herramienta indispensable en las ciencias experimentales durante mucho tiempo.

En Ingeniería en Computación, la gran diferencia radica en que el modelo no reproduce al sistema observado cuyo comportamiento deseamos conocer. El modelo se ubica en una etapa anterior. Pero nos permite evaluar si una solución, la cual está en proceso de ser descubierta, diseñada, respetará las propiedades esperadas.

En la Figura 2 representamos la relación entre las diferentes entidades de un sistema sometido a una operación de modelado. El ingeniero modela el sistema a construir mientras está en proceso de diseño. El análisis del modelo (Ej.: por simulación o métodos formales) nos permite deducir propiedades del sistema. La relación entre las propiedades del modelo y las del sistema de ninguna manera es un aspecto trivial. Es uno de los aspectos que ha dado lugar a la presentación de la idea de "*Cyber Physical Systems*" y su problemática particular.

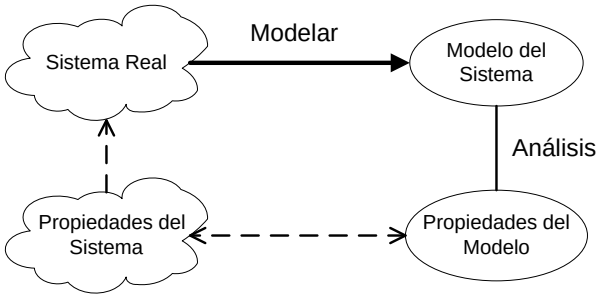

Figura 2: Relación entre el sistema, el modelo y sus propiedades.

Primero, la manera de poner de manifiesto las propiedades del sistema puede variar ampliamente en los dos casos. Habitualmente las propiedades del sistema serán expresadas en lenguaje natural, mientras que las propiedades del modelo tendrán la forma de un invariante o harán referencia con bastante precisión a una entidad de análisis (cuyo significado debe ser conocido por los usuarios si desean entenderlo). De modo que debemos anticipar elementos de trazabilidad entre ambos.

Segundo, la necesaria abstracción de ciertos elementos del sistema durante el proceso de modelado (por ejemplo, cambiar de tipos discretos a tipos continuos) puede, cuando es apropiado, resultar en un modelo que no representa fielmente al sistema real. Si el modelo resulta en un super conjunto de propiedades del sistema, entonces la propiedad que no ha sido verificada en el modelo puede ser verificada en el sistema; si es un subconjunto, entonces la propiedad que ha sido verificada en el modelo puede no ser totalmente confiable para el sistema.

Así, la tarea de modelar es una operación que presenta desafíos que deben ser manejados cuidadosamente. Concretamente, las decisiones al modelar deben ser documentadas de forma apropiada.

Por otro lado el desarrollo de CPS requiere la colaboración de múltiples actores y grupos de muy variadas especialidades. De modo que necesitamos todo el soporte necesario para garantizar una comunicación fluida entre todas las partes involucradas. Este es un muy buen rol para los modelos, los cuales son abstractos y son una representación parcial del sistema desde un punto de vista y con un nivel de granularidad que facilita el estudio de algunas de sus características (seguridad, propiedades, comportamiento, etc.).

Cada modelo es, de este modo, diseñado con un objetivo muy preciso en mente y ningún modelo es suficiente por sí mismo para traducir la complejidad propia del sistema.

UML es uno de los lenguajes de facto para el modelado de artefactos de software. SysML es un perfil de UML y es un lenguaje de modelado gráfico utilizado para capturar aspectos referidos a la estructura, funcionamiento y comportamiento del sistema, personas y procedimientos. Puede ser utilizado para dar soporte a las actividades de especificación, análisis, diseño y verificación-validación.

En el campo de los CPS, considerando el modelado de aspectos de seguridad en etapas tempranas de diseño, la herramienta CMU aplicada a nivel de sistema es coherente para construir modelos de comportamiento del sistema con el lenguaje de modelado SysML. Luego se pueden mapear los requerimientos (también modelados con SysML) de servicios de seguridad ya elicitados a nivel de sistema sobre componentes de software y hardware.

Así, la construcción de modelos que contemplen servicios de seguridad para mitigar amenazas descubiertas de forma temprana en la elicitación de requerimientos funcionales a nivel de sistema, permite la posterior construcción de una solución confiable y más segura. Sin olvidar la importancia que aporta a la trazabilidad entre las propiedades de seguridad requeridas y verificables.

4.3. Patrones de diseño

En el año 2001, en un intento por poner en ridículo el sistema de patentes, un australiano patentó un "*…dispositivo circular para facilitar el transporte…*", en realidad se trataba de la rueda, que se había reinventado nuevamente, como tantas veces se ha hecho desde hace miles de años. En una época, fuertemente influenciado por las TICs, si bien las innovaciones son conocidas globalmente de forma prácticamente instantánea, las pequeñas soluciones a problemas concretos siguen siendo olvidadas e inventadas de nuevo, una y otra vez.

Christopher Alexander nació el 1936 en Austria, aunque pasó sus primeros 22 años en Inglaterra, donde estudió arquitectura y matemáticas, mudándose finalmente a Estados Unidos donde actualmente ejerce como profesor emérito en la Universidad de California. Al mudarse empezó a aplicar sus conocimientos matemáticos y el racionalismo al diseño, culminando en la publicación del libro "*Notes on Synthesis of Form*" en 1964. Sin embargo, al cabo de unos años él mismo se corregía:

"Soy, como algunos sabréis, matemático de origen. Dediqué muchos años, en los sesenta, a intentar definir una visión del diseño aliada con la ciencia [...]. Jugué con la investigación de operaciones, la programación lineal, todos estos juguetes fascinantes, que los matemáticos y la ciencia nos ofrecían, e intenté ver cómo todo esto podía darnos una nueva visión del diseño, qué diseñar y cómo diseñarlo. Finalmente, sin embargo, observé que esta visión no era productiva."

Christopher Alexander se dio cuenta que el diseño no era algo matemático, no podía realizarse sin tener en cuenta las sensaciones y estaba muy relacionado con la vida. El diseño debía responder a necesidades de personas: un banco en un parque no era un "*banco*", era "*el banco de un grupo de personas*". Los pequeños detalles de diseño, sea en un edificio o una ciudad, debían responder a multitud de necesidades relacionadas con su entorno y, a la vez, integrarse a ese entorno. Cada una de estas necesidades tenía un modelo de solución que podía ser imitado cada vez que se detectaba el mismo problema, fuera en otro momento o en otro espacio. Este modelo replicable al que se refería Alexander, no era ni más ni menos, que un patrón de diseño.

Christopher Alexander publica en el año 1977 su libro "*A Pattern Language*", donde se puede leer su teoría de cómo el lenguaje de patrones puede ayudar a entender los objetos no como objetos aislados en sí, sino como elementos de interacción humana. Él define un patrón de diseño como:

"*…una descripción de un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar la solución un millón de veces, sin hacer lo mismo dos veces…*".

Esta descripción da un nombre al patrón; describe el problema que trata; ofrece una solución, y finalmente habla de las consecuencias, ventajas e inconvenientes, que tiene esta solución. Por ejemplo, uno de sus patrones de diseño de edificios, llamado originalmente "*South facing outdoors*", constata que la gente no usará un espacio abierto si no es soleado (exceptuando climas muy cálidos) y propone la solución de colocar siempre los espacios abiertos, como un jardín, en la parte sur del edificio. Para nuestro hemisferio sería al norte, pero la idea es totalmente aplicable.

Posteriormente los patrones fueron introducidos en el dominio del desarrollo de software por diferentes caminos. Algunos desarrolladores fueron inspirados directamente por los trabajos de Alexander. Otros

utilizaron fuentes similares, redescubriendo y reinventando aproximaciones sobre los diseños que ya se conocían. Frecuentemente se pueden encontrar soluciones similares para resolver un problema que tiene en cuenta una funcionalidad en particular, que resisten el paso del tiempo y cuyas consecuencias son muy bien conocidas.

En el libro "*Design Patterns: Elements of Reusable Object-Oriented Software*", Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides documentaron problemas de diseño que son recurrentes y las soluciones habitualmente encontradas en los frameworks orientados a objetos, escritos en C++ y Smalltalk. James Coplien documentó idiomas comunes y avanzados para C++. Kent Beck y Ward Cunningham aplicaron las ideas de Alexander directamente sobre el diseño de las interfaces de usuario en Smalltalk.

Los patrones de diseño de software proporcionan, en el diseño iterativo, las mismas ventajas que en cualquier otro ámbito y pueden ser muy útiles en cualquier proyecto ya que:

- Hacen más eficiente el diseño, pues no es necesario volver a generar alternativas para solucionar un problema y elegir una de ellas. Esto permite la reutilización del software.
- Por su antigüedad, ya han sido probados y se conoce su eficacia. Esto evita riesgos en el desarrollo de una nueva solución para un problema conocido.
- Permiten crear estándares de uso, los usuarios acaban conociéndolos e, incluso, esperándolos, lo que contribuye a un fácil aprendizaje de los sistemas.
- Crean un vocabulario común que contribuye a mejorar la comunicación interna entre diseñadores.
- Su aplicación es viable en distintas fases del proceso de diseño, desde la conceptualización hasta el diseño final, siempre y cuando se conozcan los requerimientos de usuario.

Debemos tener en cuenta en todo momento que los patrones son una herramienta de diseño entre muchas otras. De hecho, Christopher Alexander define en su libro "*The Oregon Experiment*" hasta seis principios que se deben seguir a la hora de diseñar, uno de los cuales es utilizar un lenguaje de patrones. El segundo, en lo que se refiere al diseño por interacciones, seguramente resulta familiar:

"El principio de participación: todas las decisiones sobre qué construir y cómo construirlo deben estar en manos de los usuarios."

Lo cual me animo a trasladar en estos términos:

"Todas las decisiones sobre qué funcionalidad implementar en una aplicación y cómo construirla debe estar en manos de los requerimientos". Lo cual se podría extender como corolario diciendo que *"Todas las decisiones sobre qué servicio de seguridad implementar en un sistema deben estar en manos de los requerimientos de seguridad elicitados simultáneamente con los requerimientos funcionales"*

Los patrones de diseño de software, actualmente están documentados y presentados en varios libros, artículos, sitios web (http://www.hillside.net/), conferencias PLOP (Pattern Language Of Programs) y cursos.

4.4. Patrones de seguridad

En del área de conocimiento de la seguridad del software un reconocido principio de protección es la reutilización de recursos comunes, para evitar reinventar soluciones ad-hoc desde cero. Construir una solución desde cero (por ejemplo un protocolo de cifrado) es riesgoso debido a la probabilidad de fallos en el diseño; del mismo modo no es conveniente re-implementar desde cero una solución que es bien conocida, por los defectos que puedan introducirse en la etapa de desarrollo. En este aspecto los patrones de seguridad son una ayuda para cumplir este principio de seguridad en la arquitectura y el diseño, ya que encapsulan el conocimiento experto en un formato reutilizable. Por otro lado, los patrones de seguridad deben incluir suficiente información detallada (incluso hasta el nivel de código) para ayudar a automatizar la fase de implementación. De este modo, los patrones de seguridad, se transforman en un medio adicional de proporcionar un nivel de seguridad a la construcción de software con requerimientos de seguridad.

Los principales objetivos que plantea la seguridad, son proteger la confidencialidad, integridad y disponibilidad de los datos. Los datos son un recurso muy valioso y se transforman con frecuencia en objetivo de ataques por parte de personas que desean ganar acceso a beneficios económicos, hacer un daño político o simplemente cometer vandalismo. Las contramedidas para garantizar los objetivos de seguridad se clasifican en cinco grupos [11]: Identificación y Autenticación, Control de Acceso y Autorización, Logging, Criptografía y Detección de Intrusión.

Los patrones de seguridad describen los mecanismos que caen dentro de estas categorías, o su combinación, destinados a detener o mitigar ataques, como así también los modelos abstractos que guían el diseño de estos mecanismos. Reúnen el extenso conocimiento acumulado sobre el aspecto seguridad con la estructura provista por los patrones de diseño. Juntos ofrecen lineamientos para la construcción y evaluación de sistemas seguros.

La seguridad como objeto de estudio, tiene una larga trayectoria, comenzando por los primeros modelos de Lampson [12] y Bell/LaPadula [13] en los '70 que resultaron en una variedad de aproximaciones para

analizar los problemas de seguridad y diseñar mecanismos de seguridad. La evolución que ha tenido su tratamiento, influenciada indirectamente por las ideas de Christopher Alexander, al tratar de codificar esta experiencia en la forma de patrones de seguridad, en alguna medida se podría decir que ha sido natural. Era de esperar que ocurriera.

Yoder y Barcalow escribieron el primer trabajo sobre patrones de seguridad [14]. Incluyeron una variedad de patrones útiles para resolver diferentes aspectos. Anterior a ellos, al menos tres trabajos [15, 16, 17] mostraron modelos orientados a objetos de sistemas seguros, sin llamarlos patrones o sin utilizar una de las plantillas estándares para la representación de patrones. En 1998 se documentaron dos patrones más: un patrón para criptografía [18] y un patrón para control de acceso [19]. Luego de esto, se presentaron algunos otros y hoy se pueden encontrar libros [20, 21] y otros tipos de publicaciones. Hoy los patrones de seguridad están aceptados por muchas compañías tales como Microsoft [22] y Sun [23], las que han hecho sus propias publicaciones y tienen sitios específicos en Internet. También existe un sitio general sobre patrones de seguridad [24].

5. Conclusiones

Este trabajo parte de una encuesta que presenta un diagnóstico preocupante sobre la seguridad de los sistemas embebidos que se están construyendo y que aspiran a ser parte de nuestras vidas en los productos que se comercializarán en el corto plazo en industrias como la automotriz, salud e IoT entre otras. Todos los fabricantes aspiran a conectar sus desarrollos a Internet. Sabemos que Internet es un espacio esencialmente inseguro por su omnipresencia y por haberse construido sin el debido análisis de amenazas a mitigar. También sabemos que se hacen enormes esfuerzos para mejorar esa realidad.

Sobre ese escenario es necesario cuestionarnos qué podemos hacer para no sumar una "neblina" insegura sobre la "nube" ya insegura. Qué podemos hacer cuando los sistemas embebidos evolucionan en la dirección de sistemas más complejos (Cyber Physical Systems) que aspiran a estar conectados a Internet cómo un requerimiento funcional.

Lo que proponemos son tres principios esenciales que han mostrado que pueden mejorar la seguridad de un sistema.

El primero de ellos es algo elemental. Propone que los requerimientos de seguridad de un sistema no pueden ser tratados como requerimientos no funcionales (RNF) como siempre se ha hecho. Los requerimientos de seguridad deben ser elicitados y tratados cómo requerimientos funcionales desde las primeras etapas del desarrollo ya no del software, sino del sistema que se pretenda construir. Consideramos que ese sistema estará compuesto por software y hardware, ambos procesando y tomando decisiones sobre el mundo físico.

El segundo de ellos invita a considerar la construcción de modelos para comprender detalladamente el funcionamiento de los sistemas. Podemos pensar que esto se enmarca en una estrategia de Model Driven Development (MDD). Sin embargo es posible abordar la construcción de modelos a nivel de sistema y de sus componentes de software y hardware sin pretender más que analizar detalladamente el comportamiento del sistema y fundamentalmente amenazas, para abordar su reconocimiento y el diseño de contramedidas de seguridad. Después se puede avanzar con estrategias de MDD que conducen a problemáticas más costosas en recursos. Por otro lado si consideramos que la seguridad tratada desde las etapas tempranas del desarrollo de un sistema desencadenará una serie de acciones sobre las diferentes etapas de proceso, esto supone una actividad interdisciplinaria. Los modelos contribuyen a facilitar la comunicación en este ambiente interdisciplinario.

El tercer elemento a tener en cuenta son los patrones de diseño del dominio de la seguridad. Es cierto que la adopción generalizada de los patrones de seguridad está aún rezagada, especialmente cuando se compara con el éxito que han tenido los patrones de diseño de software. Cabría preguntarse en ese contexto si ¿Existen suficientes patrones de seguridad documentados? ¿Todos los patrones de seguridad son de utilidad para la construcción de soluciones de software? ¿Están correctamente documentados? ¿Son realmente útiles en la práctica?

Hoy los patrones de seguridad tienen una cobertura del dominio de la seguridad y una documentación que mejora permanentemente. Por otro lado, si bien en algunos casos, el patrón muestra no ser una decisión realmente útil en la construcción de aplicaciones reales, contiene una enseñanza sobre la problemática resuelta que contribuye a mejorar el tratamiento de soluciones de seguridad.

Así también creemos que es relevante impulsar la utilización generalizada de patrones de seguridad, mediante la construcción de métodos y herramientas que permitan estudiarlos y aplicarlos de manera sistemática en la mejora de la seguridad de los sistemas compuestos de software y hardware.

Las experiencias que hemos hecho sobre los principios que proponemos se han llevado adelante en proyectos integradores de grado y proyectos de extensión en la carrera de Ingeniería en Computación de la Facultad de Ciencias Exactas Físicas y Naturas de la Universidad Nacional de Córdoba y publicaremos algunos resultados en futuros trabajos.

6. Referencias

[1] Critical Infrastructure Sectors; Homeland Security; https://www.dhs.gov/critical-infrastructure-sectors; consultado en Junio 2017.-

[2] 2017 Embedded Systems Safety & Security Survey; https://barrgroup.com/Embedded-Systems/Surveys/2017-embedded-systems-safety-security-survey ; Consultado en Mayo 2017.-

[3] McGregor, J., & Sykes, D. (2001); "A practical guide to testing object-oriented software". Addison-Wesley.-

[4] John Viega and Gary McGraw; "Building Secure Software: How to Avoid Security Problems the Right Way" (Boston, MA: Addison-Wesley, 2001).-

[5] Marco M. Morana (Foundstone Professional Services), "Building Security into the Software Life Cycle: a Business Case" (paper presented at BlackHat USA, Las Vegas, NV, August 2–3, 2006).-

[6] E. B. Fernandez, M.M. Larrondo-Petrie, T. Sorgente, and M. VanHilst, "A methodology to develop secure systems using patterns", Chapter 5 in "Integrating security and software engineering: Advances and future vision", H. Mouratidis and P. Giorgini (Eds.), IDEA Press, 2006, 107-126.

[7] Solinas Miguel, Trad Jairo, Abdala Juan, Capdevila Francisco, Fernandez Eduardo B., Antonelli Leandro; "Caso de éxito de método que aplica patrones de seguridad en la Ingeniería en Computación"; CACIC 2010; Ciudad de Buenos Aires, 2010.-

[8] Fabricio A. Braz , Fernandez Eduardo B. y VanHilst Michael, "Eliciting Security Requirements through Misuse Activities"; DEXA '08 Proceedings of the 2008 19th International Conference on Database and Expert Systems Application; IEEE Computer Society Washington, DC, USA ©2008.-

[9] Paco Hope and Gary McGraw (Cigital, Inc.), and Annie I. Antón (North Carolina State University), "Misuse and Abuse Cases: Getting Past the Positive," IEEE Security and Privacy (May-June 2004): 32–34. Disponible en: http://www.cigital.com/papers/download/bsi2-misuse.pdf; Accedido en Agosto 2017.-

[10] 200 Meledath Damodaran, "Secure Software Development Using Use Cases and Misuse Cases," Issues in Information Systems VII, no. 1 (2006): 150–154. Disponible en: http://www.iacis.org/iis/2006_iis/PDFs/Damodaran.pdf; Accedido en Agosto 2011.-

[11] Eduardo B. Fernandez; Hironori Washizaki; Nobukazu Yoshioka2; Atsuto Kubo; and Yoshiaki Fukazawa; "Classifying Security Patterns".-

[12] B.W. Lampson, "Protection", Procs. 5th Annual Conf. on Info. Sciences and Systs.,1971, 437-443. Reprinted in ACM Operating Systs. Review, 8, 1 (January 1974), 18-24.-

[13] E.B.Fernandez, T. Sorgente, and M.M. Larrondo-Petrie, "Even more patterns for secure operating systems', Procs. of the Pattern Languages of Programming Conference (PLoP 2006).-

[14] Yoder, J.; Barcalow, J.; "Architectural patterns for enabling application security". Procs. PLOP'97, Also Chapter 15 in Pattern Languages of Program Design, vol. 4 (N. Harrison, B. Foote, and H. Rohnert, Eds.), Addison-Wesley, 2000. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.3950&rep=rep1&type=pdf ; Accedido en Agosto 2011.

[15] E.B.Fernandez, M.M.Larrondo-Petrie and E.Gudes, "A method-based authorization model for objectoriented databases", Proc. of the OOPSLA 1993 Workshop on Security in Object-oriented Systems , 70-79.-

[16] E.B. Fernandez, J. Wu, and M. H. Fernandez, ``User group structures in object-oriented databases", Proc. 8th Annual IFIP W.G.11.3 Working Conference on Database Security, Bad Salzdetfurth, Germany, August 1994.-

[17] W. Essmayr, G. Pernul, and A.M. Tjoa, "Access controls by object-oriented concepts", Proc. of 11th IFIP WG 11.3 Working Conf. on Database Security, August 1997.-

[18] A. Braga, C. Rubira, and R. Dahab, "Tropyc: A pattern language for cryptographic object-oriented software", Chapter 16 in Pattern Languages of Program Design 4 (N. Harrison, B. Foote, and H. Rohnert, Eds.). Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.53&rep=rep1&type=pdf; Accedido en Agosto 2011.-

[Das98] F. Das Neves and A. Garrido, "Bodyguard", Chapter 13 in Pattern Languages of Program Design 3, Addison-Wesley 1998.-

[20] M. Schumacher, E.B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, Security Patterns, J. Wiley & Sons, 2006.-

[21] C. Steel, R. Nagappan, and R. Lai, Core Security Patterns: Best Strategies for J2EE, Web Services, and Identity Management, Prentice Hall, Upper Saddle River, New Jersey, 2005.-

[22] Microsoft patterns and practices development center, Disponible en: http://msdn. microsoft.com/en-us/practices/default ; Accedido en Agosto 2011.-

[23] Sun Developer Network, Disponible en: http://java.sun.com/blueprints/patterns/; Accedido en Agosto 2011.-

[24] The Security Patterns page, maintained by M. Schumacher; Disponible en: http://www.securitypatterns.org; Accedido en Agosto 2011.-