**Título: Guía metodológica para implementar la seguridad durante el desarrollo de aplicaciones informáticas**

Trabajo de Diploma para optar por el título de
Máster en Informática Aplicada


**Autora:** Ing. Ruth Yurina Vega Cutiño

**Tutores:** Dr. Walter Baluja García
Msc. Reinaldo Díaz Castro

**Consultante:** Msc. Michael González Jorrín


La Habana
Noviembre, 2011

*DECLARACIÓN DE AUTORÍA*

Declaro que soy el único autor de este trabajo y autorizo a la Dirección General de Producción de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Ruth Yurina Vega Cutiño

_____

Walter Baluja García

_____

Reinaldo Díaz Castro

_____

*DATOS DE CONTACTO*

Dr. Walter Baluja García. Decano de la facultad de Eléctrica del Instituto Superior Politécnico José Antonio Echeverría. Profesor Auxiliar. walter@tesla.cujae.edu.cu

MSc. Reinaldo Díaz Castro. Vice-Director de Tecnología de la Informática y las Comunicaciones del Complejo de Investigaciones y Tecnologías Integradas (CITI). Profesor Asistente. rdiazc@tesla.cujae.edu.cu

*A los que forman profesionales de la informática*

*y luchan por desarrollar la industria del software en Cuba*

## AGRADECIMIENTOS

Siempre que se cumple una meta en la vida hay un grupo de personas que forman parte del éxito. Quiero agradecer a todas esas personas y en especial mencionaré con los que estaré eternamente en deuda por ayudarme, apoyarme o sencillamente estar conmigo en estos momentos:

En especial a mis padres y mis amigos de siempre Laya y Yurdik, por estar no solo en este, sino en todos los momentos.

A Ariel por su cariño, por ser propulsor en el alcance de este objetivo y por su ayuda como profesional.

A Micha por su ayuda incondicional y sus sabios consejos.

A Juan Carlos por las revisiones al documento y transmitirme sus experiencias.

A mis tutores por destinarme tiempo del que no tienen.

A mis amigos y compañeros de trabajo que hacen que la vida y la rutina sea agradable: Rosy, Dainovy, Nery, Diana, Amet, Luis E., Ril, Ailec, Dayana, Jose, Keydi, Barbarita y Maité.

A los compañeros de postgrado que se esfuerzan por garantizar la superación profesional del claustro de la UCI.

A los que creen y confían en los resultados de este trabajo y colaboraron en las entrevistas y encuestas.

A los compañeros del Departamento de Seguridad Digital del CISED, que me han dado la oportunidad de ser parte de su equipo y aprender de la experiencia práctica.

A todo el claustro de profesores de Seguridad Informática de la UCI, que han colaborado en el desempeño de muchas tareas a favor de la seguridad informática y me han exigido superación, en especial al equipo de la vieja guardia: Arianne, Linnet, Miguel, Yoamel y Roberlán.

## RESUMEN

En la Universidad de las Ciencias Informáticas (UCI), la mayoría de los equipos de desarrollo de software no disponen de especialistas en seguridad. Algunos tienen la posibilidad de designar a un responsable para darle atención al tema, pero la gran parte cuenta con un número reducido de integrantes para realizar todas las tareas, por lo que dicho responsable tendrá que realizar otras actividades que le impiden alcanzar un alto nivel de conocimiento en cuanto a cómo desarrollar productos seguros.

El objetivo de este trabajo es la integración de la seguridad dentro del ciclo de desarrollo de software, sin que se pierda la simplicidad del proceso o requiera incorporación de personal adicional y especializado al equipo de proyecto. Para esto, se realizó una investigación sobre cuáles deben ser los pasos a seguir para la construcción de un producto de *software* seguro, considerando todas las etapas del ciclo de desarrollo y los principios de seguridad en el *software*.

Como resultado se obtuvo una guía metodológica, basada en los principios del manifiesto ágil y suficientemente flexible para ser aplicable a cualquier entorno de desarrollo, sin requerir de un equipo de especialistas en seguridad para poner en práctica las actividades que define. Para generalizar la propuesta, se construyó un portal temático sobre una plataforma que permite el trabajo colaborativo. En este portal, además de la información relativa a la guía para implementar la seguridad, podrán estar accesibles herramientas y complementos para desarrollar las tareas.



PALABRAS CLAVE

seguridad, ciclo de desarrollo de software, principios de seguridad, desarrollo ágil

# ÍNDICE

# TABLAS Y FIGURAS

# INTRODUCCIÓN

El desarrollo tecnológico ha propiciado la demanda de aplicaciones informáticas en gran parte de las actividades de la vida cotidiana, pero a la vez surgen nuevas amenazas a las que se exponen estos sistemas informáticos y aumenta la sofisticación de los ataques y las herramientas que se utilizan para perpetrarlos.

Para minimizar el número de ataques exitosos a los sistemas informáticos, una buena alternativa es el desarrollo de aplicaciones seguras que disminuyan la posibilidad de existencia de vulnerabilidades, pero es más fácil hallar en la red listas de debilidades junto a indicaciones y herramientas para explotarlas, que encontrar una guía práctica que ayuden a desarrollar sistemas seguros.

Se considera un producto (de software) seguro aquel que protege la confidencialidad, integridad y disponibilidad de los recursos y el servicio, bajo el control del propietario o administrador del sistema [35; 37].

No se puede afirmar que la seguridad de un software está totalmente garantizada, debido a que cada día se descubren nuevas vulnerabilidades, aparecen nuevos riesgos y se desarrollan novedosos métodos de ataque, combinando antiguas técnicas con nuevos procedimientos.

Cada aplicación tiene sus particularidades con respecto a la seguridad y a pesar de que esta es un ciclo permanente, que depende mucho de la sistematicidad en la atención que le den los programadores, el diseño inicial influye mucho en la evolución futura como producto seguro.

"(…) la seguridad necesita ser un componente íntegro de cada fase de la vida de una aplicación y no un complemento de *software*. A pesar de que las metodologías de desarrollo de *software* toman acciones en este sentido, comúnmente no se le da la atención necesaria a la seguridad durante todo el ciclo de desarrollo de *software"* [30].

Las metodologías conocidas como "pesadas" o tradicionales, hacen énfasis en la planificación y control del proyecto y en la especificación precisa de requisitos y el modelado; por lo que imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de obtener aplicaciones informáticas que se ajusten a los requisitos del negocio que solicita cliente. Esto hace necesario planificar un mínimo de

seguridad durante la concepción del sistema; aun así, en la práctica es insuficiente el tratamiento que se le da a la seguridad.

Las metodologías ágiles, tienen como objetivo hacer las entregas del proyecto menos complicadas, obviando un poco algunos detalles de la documentación y sin tener a la seguridad como la prioridad fundamental [48; 51]. Pero ya que los métodos ágiles se centran en crear soluciones rápidas que satisfagan las necesidades de los clientes y la seguridad es una necesidad del cliente, hay que tomar acciones concretas para proporcionarle la debida atención. En lo últimos años, los métodos ágiles han adquirido un auge significativo y específicamente en los centros de desarrollo de la Universidad de las Ciencias Informática (UCI), se está poniendo en práctica la explotación de las ventajas de estas metodologías desplazando a las tradicionales en la mayoría de los proyectos de desarrollo.

Al comienzo de esta investigación, se realizaron entrevistas a líderes de proyectos de *software* y se aplicó una encuesta con el objetivo de diagnosticar la situación del entorno; para ello se seleccionó una muestra de diez proyectos de diferentes facultades de la UCI. En las entrevistas se identificó falta de conocimiento respecto a los temas de seguridad y un uso prácticamente nulo de estándares, modelos o esquemas para atender la seguridad en el desarrollo de proyectos de *software*, solo algunos centros que trabajan muy de cerca la seguridad tienen en cuenta algunos de los elementos más conocidos. Otra característica presente en los proyectos encuestados es la falta de personal entrenado en temas de seguridad, unido a que los grupos de desarrollo son reducidos en número de profesionales y no dan a abasto con las funciones que hay que cumplir. Al procesar los resultados de la encuesta, las estadísticas no coinciden con los problemas detectados durante las entrevistas, lo que significa que no solo hay desconocimiento, sino que tampoco se reconoce la existencia de los problemas y por consiguiente la necesidad de darles solución.

Sólo una pequeña parte de los grupos de desarrollo, tienen en cuenta algunas de las buenas prácticas de seguridad para garantizar los requerimientos de calidad y el funcionamiento del *software.* Mayormente lo hacen sin realizar un análisis previo profundo, sino más bien guiados por las facilidades de las tecnologías o metodologías de desarrollo que emplean, tendencias de otros grupos o por su propia intuición.

En el proyecto ERP-Cuba se desarrolló una vista de arquitectura de seguridad [13], que rige los procesos de codificación, comunicación y configuración basados en

estándares existentes. Este es un primer intento para pautar los pasos a seguir y conseguir una arquitectura de seguridad; pero no es suficiente, definir los estándares de codificación para el desarrollo de un sistema informático es solo un punto a tener en cuenta para obtener un producto seguro.

En la Facultad 2 se desarrolló una Plataforma de Gestión de Pruebas de Seguridad Informática, con módulos específicos para distintos propósitos. El módulo de pruebas de intrusión [10] por ejemplo, se basa en las consideraciones de la guía de prueba de OWASP *(The Open Web Application Security Project)*, lo que evidencia que es una solución bien fundamentada y constituye otro intento por lograr la obtención de aplicaciones seguras. Esta plataforma cuenta con otros módulos [11], que también son excelentes trabajos, al igual que el procedimiento para pruebas realizado en la Facultad 4 [12]. No obstante a los esfuerzos realizados, la práctica de pruebas de manera aislada, no resuelve el problema.

En las plantillas de proyecto técnico que se emplean en la UCI, no se exigen detalles relativos al desarrollo de la seguridad en los proyectos, ejemplo, no se solicita entre los artefactos los casos de abuso[1], o al menos el resultado del análisis de riesgos de la aplicación informática como un entregable [1].

En la UCI se trabaja con una guía base para el diseño arquitectónico, en que se describen las nueve vistas de la arquitectura agrupadas en tres vistas principales: Vistas de Arquitectura de Sistema, Vistas de Arquitectura de Tecnología y Vistas de Arquitectura de Infraestructura. La Vista de Seguridad se encuentra dentro de las Vistas de Arquitectura de Tecnología. En dicha vista, la seguridad de las aplicaciones se establece en profundidad y por niveles, esta es una excelente iniciativa, pero solo para la auto-evaluación de los proyectos pues solo incluye que hacer, no como hacerlo. En el dictamen sobre el estado de elaboración del artefacto: Diseño de la Arquitectura de *Software*, se tienen en cuenta los elementos más significativos de la Vista de Seguridad [9].

Uno de los mejores métodos para evitar la aparición de *bugs* de seguridad en aplicaciones en producción es mejorar el Ciclo de Vida de Desarrollo del Software (en ingles *Software Development Life Cycle,* o *SDLC*) incluyendo la seguridad [60].

Integrar prácticas de ingeniería seguras en todo el ciclo de vida del *software* permite

[1] Ver glosario de términos

alcanzar una garantía de seguridad de aplicaciones mediante un enfoque global. El proceso requiere una participación directa con desarrolladores y evaluadores de *software*, ser incremental, iterativo y debe personalizarse para cada organización [2].

En el área internacional, existen algunos proyectos y trabajos encaminados a desarrollar manuales de consulta, buenas prácticas y metodologías para el desarrollo de aplicaciones seguras.

Tal es el caso de los proyectos de *The Open Web Application Security Project (OWASP)*, enfocados a la seguridad de aplicaciones Web [59; 61]. También existe la metodología de Modelado de Amenazas de Microsoft y otras estrategias como *Trike*, *CVSS* y *Octave*, entre otras [4; 41; 61].

El estándar *SPSMM (Secure Programming Standards Methodology Manual)*, forma parte del manual *OSSTMM (The Open-Source Security Testing Methodology Manual).* Con *OSSTMM* se descubre que hay un error y SPSMM descubre donde está ubicado, cómo arreglarlo y evitar su aparición en futuras codificaciones [45; 63].

Otra iniciativa es el proyecto *Build Security In (BSI),* un intento por organizar la información referida a la seguridad del *software* en un sitio Web para mantenerla fácilmente accesible [56].

Microsoft también propone una metodología, *SDL (Security Development Lifecycle),* la cual permite que la seguridad se introduzca en el desarrollo del *software* desde las etapas iniciales. Esta metodología está adaptada específicamente a las condiciones de *Microsoft* [37].

A pesar de estas iniciativas existentes, adoptar su aplicación como solución se hace difícil, dado que algunas están diseñadas para ser aplicadas en entornos con características muy particulares y diferentes al modelo de la UCI o ajustadas a una metodología de desarrollo de *software* específica, otras requieren de un equipo altamente cualificado en temas de seguridad o están enfocadas hacia un único objetivo de la seguridad.

Producto de la problemática existente se ha formulado el siguiente **problema a resolver:** "¿cómo definir un proceso flexible, ágil y de fácil aplicación a través de todo el ciclo de desarrollo del *software*, que permita la construcción de un producto seguro sin requerir especialistas en seguridad o incluir nuevos miembros al equipo de desarrollo para ponerlo en práctica?"

Para ello se definió como **objeto de estudio** el ciclo de vida de desarrollo del software y como **campo de acción** la seguridad integrada al ciclo de desarrollo del software.

Cómo **hipótesis** de la investigación se plantea: Si se elabora una guía metodológica ágil, basada en un flujo de tareas a realizar para obtener un *software seguro,* teniendo en cuenta todas las etapas del ciclo de desarrollo y que no requiera especialistas en seguridad para su puesta en práctica, se puede fortalecer la seguridad de los productos de *software* desarrollados en la UCI.

El **objetivo general** de la investigación es construir una guía metodológica que indique cómo lograr el desarrollo de aplicaciones seguras mediante un proceso ágil y teniendo en cuenta todas las etapas del ciclo de desarrollo del *software*.

Los métodos de investigación que se han empleado a través de todo el trabajo pertenecen tanto al grupo teórico como al empírico [50]. Del primer grupo se empleó el método inductivo-deductivo para el estudio que posibilitó la confirmación de la hipótesis y la concepción de la guía.

Por otra parte, los métodos empíricos utilizados fueron: observación y encuesta. La observación para el planteamiento del problema y la hipótesis; la encuesta para la recolección de información.

La novedad de los resultados de la tesis radica en la obtención de una guía de referencia para implementar aplicaciones informáticas seguras., a partir del estudio de las principales razones de fallas de seguridad del *software* y los principios, procesos y metodologías existentes. Dicha guía está estructurada de una forma genérica e integrable a cualquier ciclo de desarrollo.

Se obtuvo también un manual de utilidad práctica, que responde a la propuesta de la guía metodológica. Está confeccionado de forma que los detalles del modelo y organización del desarrollo de proyectos es transparente, por lo que es suficientemente flexible para ser aprovechado por otras instituciones que se dedican al desarrollo de *software*.

El resultado de esta investigación es una contribución importante, que al ponerla en práctica mejorará significativamente el proceso de desarrollo de la seguridad de los proyectos de *software* en la UCI y su adopción aportará una visión de las posibilidades y alternativas que los equipos de desarrollo disponen para la producción de *software* seguro.

Para mostrar el desarrollo de la investigación y los resultados, el trabajo se ha estructurado en tres capítulos, además de las Conclusiones, Recomendaciones, Bibliografía y Anexos.

En el primer capítulo se muestra la síntesis de la revisión bibliográfica y se describe la base teórica sobre la que se sustenta el trabajo. Se caracterizan guías y estándares internacionales referentes a la seguridad en el desarrollo de aplicaciones informáticas y de algunas de las metodologías para el análisis y gestión de riesgos.

En el segundo capítulo se presenta la propuesta de solución al problema científico. Se describe la estructura y contenido de la guía metodológica, así como el fundamento teórico de las decisiones más importantes asumidas para definir los pasos incluidos dentro de cada etapa y los artefactos de salida.

En el tercer capítulo se describen la evaluación de la solución empleando el método de criterio de experto con varias iteraciones y sin intercambio directo.

# CAPÍTULO 1. TENDENCIAS DEL DESARROLLO DE SOLUCIONES PARA LA CONSTRUCCIÓN DE *SOFTWARE* SEGURO

## Introducción

El problema de la mayoría de los desarrolladores de *software* radica en que la seguridad no es vista como una función generadora de ganancias del proceso de desarrollo [35].

Se considera un producto (de software) seguro aquel que protege la confidencialidad[2], integridad[2] y disponibilidad[2] de los recursos y el servicio, bajo el control del propietario o administrador del sistema [35; 37].

Una vulnerabilidad es una falla en el producto que hace imposible prevenir cualquier ataque o mal funcionamiento, desde la usurpación de privilegios del usuario del sistema y comprometimiento de datos, hasta asumir derechos no otorgados [24; 37].

En la propuesta de *Howard* y *LeBlanc* en su libro *Writing Secure Code* [35] se critican algunas falsas razones por las cuales las personas no construyen productos seguros:

- La seguridad es tediosa.
- La seguridad es difícil de evaluar.
- La seguridad usualmente no es la primera habilidad o interés de los diseñadores y desarrolladores en la creación del producto.
- La seguridad implica no hacer algunas cosas interesantes y nuevas.
- La seguridad es vista como una funcionalidad independiente, como algo que se toma en el camino.

Sin embargo, la verdadera razón por la cual no se construye *software* seguro primero que todo es, la falta de conocimiento por parte de los programadores sobre el tema y la falta de control de la calidad más allá del chequeo de la funcionalidad del *software*.

"Al terminar los estudios universitarios o profesionales sobre informática y disciplinas afines, por lo general no disponen de los conocimientos necesarios para comenzar a trabajar en un equipo destinado al diseño, el desarrollo o la prueba de *software* seguro.

[2] Ver glosario de términos

En general, los diseñadores de *software*, los ingenieros y los encargados de las pruebas también carecen de los conocimientos adecuados sobre seguridad [36]."

Las vulnerabilidades de la mayoría de las aplicaciones se deben a arquitecturas inadecuadas o deficiencias en su diseño. No obstante, la seguridad de las aplicaciones debe ser tratada en las diferentes etapas del ciclo del desarrollo del *software* o *SDCL* (Por sus siglas en inglés *"Software Development Life Cycle Process"*), abarcando: diseño, implementación, prueba y despliegue [37].

## 1.1.Seguridad integrada al ciclo de vida del software

El estándar ISO/IEC 12207 de la IEEE [18] agrupa las actividades que pueden ser desarrolladas durante el ciclo de vida del *software* en cinco procesos primarios: Adquisición, Abastecimiento, Desarrollo, Operación y Mantenimiento; ocho procesos de soporte: Documentación, Manejo de la configuración, Garantía de calidad, Verificación, Validación, Revisión conjunta, Auditoría, Resolución de problemas y cuatro procesos de organización: Gestión, Infraestructura, Mejora y Entrenamiento [18].

El estándar describe los procesos que son aplicables a lo largo del ciclo de vida del *software*, sin embargo, puede usarse de diferentes maneras, con diferentes vistas y objetivos de acuerdo a las características del proyecto y la organización donde se aplique [18].

Según la norma, el proceso del *software* se compone de 17 grupos de actividades que son responsables de satisfacer los requisitos asociados al grupo al que pertenecen. El proceso se compone de un total de 65 actividades.

No importa qué modelo de ciclo de vida se siga, lo fundamental es que la seguridad sea considerada desde las fases tempranas del ciclo de vida de *software*.

A partir de la definición del estándar *IEEE/EIA* 12207 se puede establecer un proceso adaptado a las características del entorno, que defina el desarrollo de la seguridad para aplicaciones informáticas.

### 1.1.1. Requerimientos

Un requerimiento se define como una propiedad que debe ser mostrada con el fin de resolver algunos problemas del mundo real [16].

Los requerimientos para las funcionalidades de seguridad en sistemas de *software* no es lo mismo que requerimientos para el *software* seguro. Los primeros tienen en cuenta las funciones que ponen en práctica una política de seguridad y se conocen también como 'requerimientos de servicios de seguridad. Los requerimientos para el *software* seguro tienen como objetivo reducir o eliminar las vulnerabilidades en el sistema y asegurar que se mantendrá funcionando correctamente incluso cuando se encuentre bajo amenaza. Estos se conocen también como objetivos de seguridad [8; 23].

De las siete áreas del conocimiento de los requerimientos de s*oftware*: Principios, Requerimientos de s*oftware*, Proceso de requerimientos, Captura, Análisis, Especificación, Validación y las Consideraciones prácticas; las más importantes a tener en cuenta para manejar los requerimientos de seguridad, son el proceso de análisis y la validación.

El proceso de análisis de requerimientos se realiza para detectar y resolver conflictos entre los requisitos y descubrir los límites del software y la interacción con el entorno. Incluye la clasificación de requisitos, el modelado conceptual, diseño arquitectónico y negociación de los requerimientos.

La validación tiene que ver con el proceso de examinar los requerimientos documentados para asegurarse de que están definiendo el sistema que el usuario espera y en las consideraciones prácticas se describe la naturaleza iterativa del proceso de requisitos, así como la gestión de cambios y el mantenimiento de los requisitos en un estado que refleja con precisión el *software* a construir.

### 1.1.2. Arquitectura y diseño

La arquitectura y el diseño de la aplicación deben analizarse desde una perspectiva de seguridad. La documentación de la etapa de diseño puede ayudar, es ese el momento en que se descompone la aplicación y se determinan los elementos importantes, como por ejemplo, el flujo de información o los puntos de partida [30].

Durante el proceso de concepción de la arquitectura y diseño de la aplicación debe tenerse en cuenta [22]:

- La planificación del despliegue y la infraestructura, teniendo en cuenta las restricciones de seguridad impuestas por la infraestructura de la capa sub-

yacente, el impacto de los servidores de la capa de aplicación intermedia, la zona perimetral y el cortafuego interno en el diseño que se propone, así como el ambiente del despliegue designado y las políticas de seguridad asociadas.

- El modelado de amenazas seleccionado para repasar cada área crítica de la aplicación, entre las que están la autorización, autenticación, validación de la entrada de datos y manejo de excepciones [57].

### 1.1.3. Implementación

Antes de comenzar la etapa de implementación es necesario llevar a cabo una revisión de la arquitectura y diseño de la aplicación en todos los aspectos relacionados con la seguridad, incluyendo un análisis de cada nivel lógico de la aplicación; ejemplo, código de acceso a datos, páginas Web dinámicas, controles y servicios Web. La figura 1 muestra los tres aspectos fundamentales en el proceso de revisión de la arquitectura y diseño de la aplicación desde la perspectiva de la seguridad [38].

Existen algunas consideraciones a tener muy en cuenta para una correcta implementación, sea cual sea el método que se seleccione, entre las que se destacan: validación de entradas, autenticación, autorización, manejo de la configuración, manipulación de información sensible, manejo de sesiones, criptografía, manipulación de parámetros, manejo de excepciones y auditoría y registro de eventos [29; 30; 49].

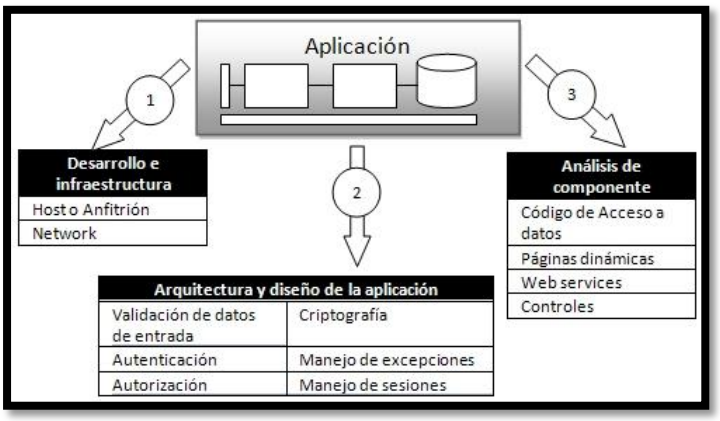


**Figura 1. Proceso de revisión de la arquitectura y diseño de la aplicación [30].**

Algunas guías y manuales pautan buenas prácticas y elementos a tener en cuenta para el desarrollo de aplicaciones seguras, entre las propuestas se pueden citar los estudios de Michael Howard, David LeBlanc, Steve Lipner y Kenneth van Wyk [30; 35; 37].

### 1.1.4. Pruebas

El objetivo principal de la prueba y análisis de la seguridad de *software* es la comprobación de que el *software* exhibe las propiedades y conductas deseables siguientes [33; 60]:

- Su comportamiento es predecible y seguro.
- No expone ninguna vulnerabilidad o debilidades
- Las rutinas que se ocupan de los errores y excepciones le permite que mantenga un estado seguro frente a modelos de ataque o fallas intencionales.
- Satisface todos los requisitos no funcionales de seguridad especificados e implícitos
- No viola ninguna restricción de seguridad especificada
- El código ha sido disimulado para evitar la ingeniería inversa

Las pruebas tienen especial importancia y para ello existen varios intentos por normar el proceso para comprobar la seguridad. Por ejemplo el *Open-Source Security Testing Methodology Manual* [45] y el *Security Programming Standard Methodology Manual* [63].

#### *1.1.4.1. Open-Source Security Testing Methodology Manual (OSSTMM)*

El OSSTMM, por sus siglas en inglés, es un estándar de código abierto para comprobar la seguridad. Describe un conjunto de reglas y líneas base para realizar test de penetración o hacking ético y define una lista de parámetros a comprobar y un orden de flujo a seguir, en ocasiones, corriendo más de una prueba en paralelo.

Esta metodología es un modelo de estándar que aunque su objetivo sea descubrir la presencia de un error, puede ser un buen apoyo para barrer los parámetros a tener en cuenta y conseguir cierto nivel de seguridad en la programación de aplicaciones Web, previniendo la ocurrencia de errores [45].

#### *1.1.4.2. Security Programming Standard Methodology Manual (SPSMM)*

El SPSMM es un estándar de programación segura creado en el "Institute for Security and Open Methodologies" (ISECOM) y forma parte además, del manual OSSTMM, por tanto, emplea la misma metodología. Sin embargo, su objetivo está en ubicar y saber cómo arreglar un error una vez que se ha descubierto y sólo es factible para partiendo de lo que se plantea, establecer lineamientos y/o procedimientos para la programación segura [63].

## 1.2.Desarrollo ágil

Los métodos ágiles son caracterizados por el logro de satisfacción del cliente a través de las entregas tempranas y frecuentes de *software* factible y utilizable (iteraciones cortas), el desarrollo reiterativo, aceptación de cambios de los requisitos finales, integración con el cliente en el ambiente de desarrollo, desarrollo paralelo de múltiples entregables, auto-organización de los equipos de desarrollo [48; 51].

Según *Philippe Kruchten y Konstantin Beznosov* [26], los métodos ágiles son de naturaleza iterativa, no siguen el tradicional método lineal: requisitos-diseño-implementación-prueba. En cambio, repiten la sucesión de este desarrollo "tradicional" muchas veces. De esta manera, los métodos ágiles pueden asemejarse al modelo espiral, pero también tienen un acercamiento al modelo evolutivo de la producción de *software* con varias actividades del ciclo de vida que ocurren al mismo tiempo.

Los métodos ágiles evitan las actividades que [51]:

- No involucren directamente la producción de *software*
- No puedan ser realizadas por miembros del equipo de *software* y concurrentemente con otras actividades del ciclo de vida.
- Requieran conocimientos especializados más allá de lo esperado de los desarrolladores en el equipo de *software*. Los métodos ágiles no permiten la inclusión de expertos en seguridad u otro personal en los equipos, que no sean los desarrolladores de *software*.
- No se enfoquen en el objetivo de producir *software* de forma rápida y correcta. Los proyectos ágiles tienen dificultades para incorporar objetivos no funcionales tales como fiabilidad y seguridad.

- Tengan que ser realizadas en un ambiente con restricciones: de quien trabaja en el proyecto, en qué rol y bajo qué condiciones de trabajo.

### 1.2.1. Tratamiento de la seguridad en los métodos ágiles

La mayoría de los requerimientos y buenas prácticas de seguridad pueden ser fácilmente incorporadas a los métodos ágiles [26]. Sin embargo, solo uno de los trece principios del manifiesto ágil tiene implicaciones positivas en la seguridad.

Algunas de las técnicas que permiten los métodos ágiles sirven de “malla de seguridad” en el desarrollo del *software* [26]:

- Refactorización (*refactoring*): Cambios en el diseño sobre el sistema implementado.
- Pruebas automáticas: Pruebas exhaustivas del sistema cuyos resultados se comparan con resultados esperados.
- Integración continua: Proceso iterativo de creación de código, que permite que en todo momento esté listo para la entrega o, en el caso de que contenga algún error, su detección es inmediata.

### 1.2.2. Roles

La definición de los roles es particular de cada metodología, pero todas convergen en que los equipos son de un número pequeño de integrantes.

En algunas metodologías los equipos son multifuncionales, cada miembro tiene un rol específico pero se le permite realizar otras funciones [25]. En otras, el equipo trabaja en conjunto, cada miembro tiene un rol definido y adquiere relevancia en las distintas etapas del proceso de desarrollo. Los roles se pueden combinar teniendo en cuenta que algunas ‘‘mezclas’’ conllevan riesgos [28].

## 1.3.Procesos y metodologías para el desarrollo de la seguridad

Existen metodologías que modifican actividades tradicionales de los SDLC, o insertan nuevas actividades con el objetivo de reducir el número de debilidades en el *software* ante el creciente número de amenazas. Algunas metodologías han sido usadas satisfactoriamente en proyectos de desarrollo de productos o en proyectos pilotos académicos. Se pueden destacar: *Microsoft Trustworthy Computing Security Development Lifecycle [36]; Oracle Software Security Assurance Process [44]*;

*Comprehensive, Lightweight Application Security Process (CLASP) [58]* y *Team Software Process for Secure Software Development - TSP-Secure* [20].

### 1.3.1. Microsoft Trustworthy Computing SDL

La metodología Microsoft *Trustworthy Computing Security Development Lifecycle* (MTC SDL), establecida formalmente por Microsoft para su proceso de desarrollo de *software* mejorado en seguridad, está diseñada de manera que pueda ser empleada en un sentido amplio [36].

Se compone de las seis fases descritas en la tabla 1 [36], y es un medio para modificar los procesos de desarrollo tradicionales, integrando tareas y puntos de chequeo expresamente destinados a la mejora de la seguridad del *software*. Sus metas son dos:

- Reducir el número de defectos de diseño y programación relacionados con la seguridad.
- Reducir la gravedad del impacto de los defectos residuales.

**Tabla 1. Fases de *Microsoft Trustworthy Computing SDL* [36].**

| Fase | Descripción |
|---|---|
| **Requerimientos** | El equipo revisa como la seguridad puede ser integrada dentro del proceso de desarrollo, identifica los objetivos críticos y considera como las características de seguridad pueden afectar o ser afectadas probablemente por otro *software* al ser usados al mismo tiempo. |
| **Diseño** | Los participantes clave, arquitectos, desarrolladores y diseñadores, ejecutan tareas de diseño, incluyendo especificaciones que detallan los elementos técnicos de la implementación. |
| **Implementación/ Desarrollo** | El equipo de producto programa, prueba e integra el *software*. Los desarrolladores usan herramientas de seguridad y listas de chequeo de la seguridad, así como, buenas prácticas de programación segura. |
| **Verificación** | El *software* es completamente funcional y entra en fase beta. El equipo de producto puede llevar a cabo pruebas de penetración en este punto para ofrecer garantías adicionales de que el *software* será resistente a las amenazas después de su lanzamiento. |
| **Liberación** | *Software* está sujeto a una revisión de seguridad final. El equipo de seguridad central evalúa el *software* antes de su envío y puede hacer preguntas de verificación de seguridad en un cuestionario. La revisión final de seguridad (FSR) también prueba la capacidad del *software* para soportar nuevas denuncias de vulnerabilidades que afectan a productos de *software* similares. |
| **Soporte y servicio** | El equipo de producto lleva a cabo cualquier evaluación post-producción necesaria, reporta vulnerabilidades y toma acciones oportunas, como la actualización del proceso SDL, la instrucción y el uso de herramientas. |

El ciclo de vida de desarrollo de seguridad (*SDL, Security Development Lifecycle*) de *Trustworthy Computing* supone que hay un grupo central en la compañía (u organización de desarrollo de *software*) que controla el desarrollo y la evolución de las prácticas recomendadas de seguridad y las mejoras de los procesos, este equipo de seguridad debe ser altamente cualificado y estar disponible para recurrir a él con frecuencia durante el diseño y el desarrollo del *software* [36].

La adopción de esta solución no es viable, si se tiene en cuenta que una de las restricciones en la mayoría de los proyectos de desarrollo en la UCI la imponen los recursos humanos. Por un lado la falta de personal especializado y capacitado en temas de seguridad y por el otro, el reducido número de integrantes con que cuentan los proyectos para ejecutar todas las tareas, incluyendo la formación de estudiantes.

### *1.3.2. Oracle Software Security Assurance Process*

*Oracle Corporation* sigue un proceso que obliga a cumplir las normas de codificación segura y el uso de bibliotecas de funciones estándar de seguridad (autenticación, cifrado, etc.). Para realizar pruebas de seguridad, se emplea la exploración automatizada de vulnerabilidades y validaciones contra listas de control de seguridad [44].

A pesar de sus ventajas, Oracle no considera su *Software Security Assurance Process* recomendable para ser ampliamente utilizable o adaptable por las empresas de *software*. Puede servir de orientación para los consumidores de sus productos, principalmente en apoyo de la configuración de bases de datos de seguridad, evaluación de vulnerabilidades y la gestión de parches.

### *1.3.3. Team Software Process-Secure*

*CMU's SEI and CERT Coordination Center (CERT/CC)* desarrollaron *Team Software Process-Secure (TSP-Secure),* cuyos objetivos son reducir o eliminar las vulnerabilidades de software que resultan del diseño y errores de aplicación, además de proporcionar la capacidad para predecir el riesgo de vulnerabilidades en el software [20].

*TSP-secure* está diseñado para guiar a través de la ingeniería de procesos, reduciendo la probabilidad de que inadvertidamente se omitan pasos, se organicen estos en un orden improductivo, o pasar tiempo innecesario averiguado el siguiente movimiento.

Para poder emplear *TSP* el equipo de desarrollo tiene que entrenarse en *PSP (Personal Software Process).*

### 1.3.4. Otros modelos en investigación y desarrollo

Existen modelos desarrollados por investigadores académicos, que se han puesto a prueba solo en proyectos pilotos, por lo que no se consideran completos y todavía se está trabajando en su refinamiento o modificaciones.

*Effective Guidance in Information Security* (*AEGIS*) surgió dentro de la comunidad de desarrollo de *software* cuando los diseñadores observaron las múltiples complejidades para lograr el plan de seguridad del sistema a lo largo del ciclo de vida al emprender proyectos con requisitos contradictorios, como la funcionalidad, la utilidad, la eficacia y la simplicidad. Cada uno de esos atributos tiende a competir con los otros, y con las metas de seguridad del sistema. AEGIS usa un modelo de desarrollo del *software* en espiral integrado a la seguridad y usabilidad con UML [19].

Este modelo tiene cuatro sesiones de diseño en las que se realizan las siguientes actividades: Identificar los activos y requerimientos de seguridad; Análisis de riesgos y diseño de la seguridad; Identificar riesgos, vulnerabilidades y amenazas al sistema.

*Rational Unified Process-Secure (RUPSec),* son extensiones de seguridad de la metodología RUP y propone nuevos artefactos de seguridad para construir los casos de uso [42].

*Secure Software Development Model (SSDM)* integra la seguridad en el proceso de desarrollo de *software* a través del uso de un modelo unificado que comprende varias técnicas existentes por producir el *software* seguro. Define un flujo de trabajo de cuatro fases dentro del proceso de diseño de *software*: Entrenamiento, Modelado de amenazas, Especificaciones de seguridad y Revisión de las especificaciones de seguridad [54].

Con la excepción del *Microsoft's Security Development Lifecycle (SDL)* y *Oracle's Software Security Assurance Process*, no hay evidencia documentada de que las metodologías existentes de desarrollo de software seguro, se hayan usado en

proyectos reales de otras instituciones. A pesar de que la efectividad de estos modelos no ha sido probada, el estudio de sus características resultó ser un interesante aporte para la definición de las etapas y tareas que debe contener un ciclo de desarrollo de la seguridad para aplicaciones.

Otra razón por la cual ninguno de estos modelos puede servir de solución, es que no se basan en los principios del desarrollo ágil. La mayoría se corresponden con procesos sumamente engorrosos, que complejizan el desarrollo y extienden demasiado los cronogramas.

## 1.4.Siete puntos de contacto para la seguridad del software

McGraw define siete puntos de contacto para el desarrollo de la seguridad del software [15], que se muestran en la figura 2.

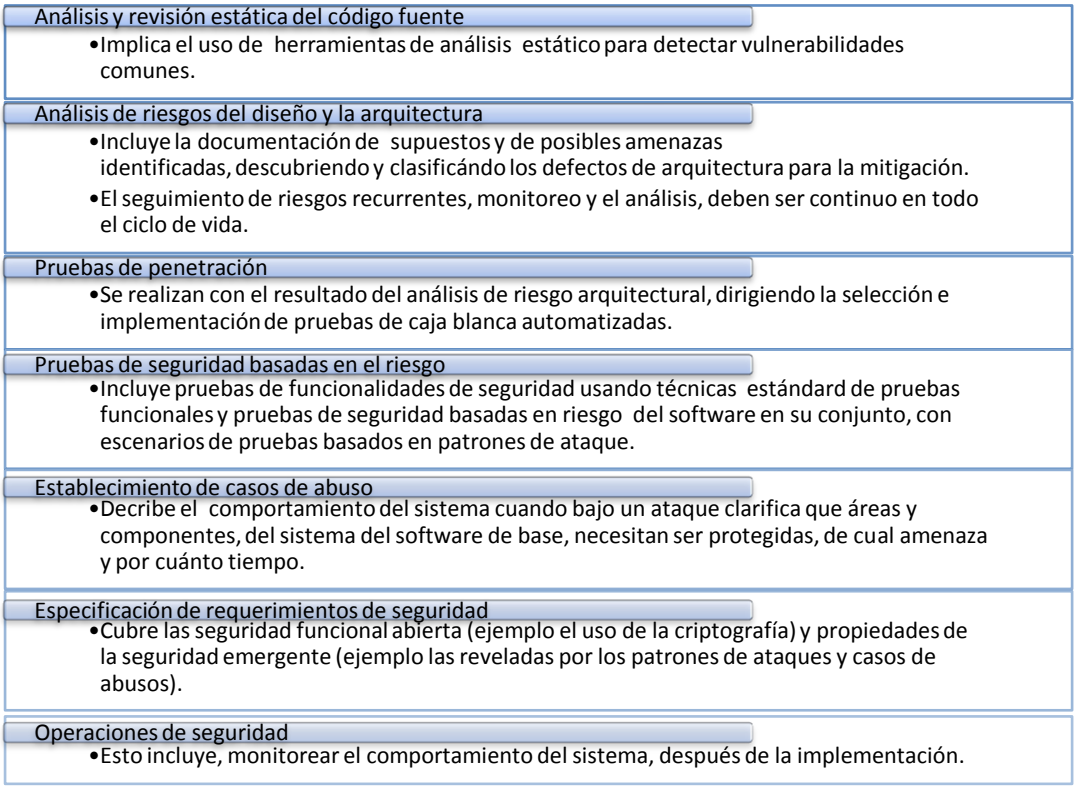


**Figura 2. Siete puntos de contacto para la seguridad del software [15]**

Estos puntos de contactos son buenas prácticas que pueden ser aplicadas a varios artefactos de desarrollo de software y no están limitadas para un ciclo de desarrollo en particular.

El Ciclo de Vida de Desarrollo de la Seguridad es un ciclo de vida de software normal, que contiene las nuevas e importantes subdivisiones de seguridad entre las fases teniendo en cuenta los siete puntos de contacto, como muestra la figura 3.

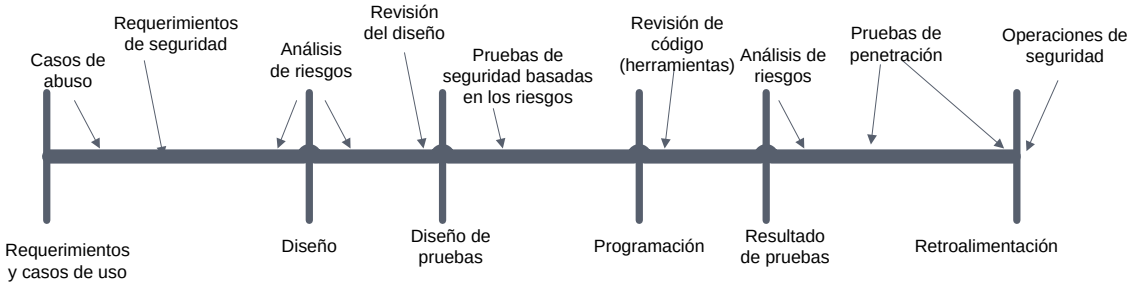


**Figura 3. Ciclo de Vida de Desarrollo de la Seguridad [15].**

Han surgido otras variantes [8; 17; 37; 38] para definir el ciclo de vida de desarrollo de las seguridad, pero en todas se evidencia la extensión del ciclo de vida de desarrollo del software a partir de las buenas prácticas que propone McGraw.

## 1.5.Principios para el desarrollo de software seguro

Un punto importante en el que coinciden la mayoría de los expertos, es en algunos principios de alto nivel para la construcción de *software* más seguro [6; 29; 37].

*Secure Windows Initiative Tea* [34] los refiere como SD3+C [36]:

- **Seguro por Diseño**: El *software* tiene que ser estructurado, diseñado e implementado para a resistir los ataques y protegerse él mismo y la información que este procesa.
- **Seguro por Defecto**: Para minimizar el daño cuando un atacante hace blanco en alguna vulnerabilidad, el estado predeterminado del *software* debe elegir las opciones más seguras.
- **Seguro en el Despliegue**: Se debe incluir con el *software* información y herramientas que ayuden a los administradores y a los usuarios a utilizar este *software* con seguridad. Las actualizaciones deben ser fáciles de distribuir y

debe existir un sistema de respuesta relativamente rápido frente a la aparición de nuevas amenazas.

- **Seguro en las Comunicaciones**: El equipo de desarrollo debe estar preparado para detectar las vulnerabilidades de seguridad del producto y comunicarse de manera abierta y responsable con los usuarios y los administradores para ayudarles a tomar las medidas de protección adecuadas (como la actualización o la implementación de soluciones alternativas).

Cada uno de los elementos anteriormente descritos impone requisitos en el proceso de desarrollo, pero los dos primeros, *Seguro por diseño* y *Seguro por defecto*, proveen la mayoría de los beneficios de seguridad. *Seguro por diseño* prevé la introducción de vulnerabilidades desde el primer plano, mientras *Seguro por defecto* garantiza que de ocurrir un ataque, por defecto la exposición del *software* sea mínima [36].

*Saltzer and Schroeder* también identifican un conjunto de principios de control de acceso y protección para tener en cuenta durante el diseño de un sistema seguro: economía del mecanismo, fallar de forma segura, mediación completa, diseño abierto, separación de privilegios, menor privilegio, menor mecanismo común y la aceptación psicológica [21].

*Howard y LeBlanc* proponen: aprender de los errores, minimizar la superficie de ataque, defensa en profundidad, asumir que los sistemas externos son inseguros, contar con un plan de fallo, no depender solo de la seguridad por oscuridad, no mezclar código y datos, así como, corregir los problemas de seguridad correctamente [35].

Para proponer cualquier solución, con la intención de obtener software seguro, deberá estar basada en estos principios.

## 1.6.Modelado de amenazas

Aplicar un Modelado de Amenazas (*Threat Model-TM*) que conduzca el diseño es el aspecto más importante en el proceso de desarrollo del *software* desde el punto de vista de la seguridad, aunque esta no es la única metodología para diseñar sistemas seguros, pues existen otras herramientas, como el empleo de buenas prácticas y arquitecturas seguras.

El modelado de amenazas es una técnica de ingeniería cuyo objetivo es ayudar a identificar y planificar de forma correcta la mejor manera de mitigar las amenazas de una aplicación o sistema informático.

### 1.6.1. *Microsoft Threat Modeling Process*

Como propuesta de un proceso de modelado de amenazas para el *software* vale mencionar la que formulan *Michael Howard* y *Steve Lipner* [37] y el modelo de la iniciativa *Trustworhty Computing con cuatro pasos que se ilustran en la* Figura 4 [38]:

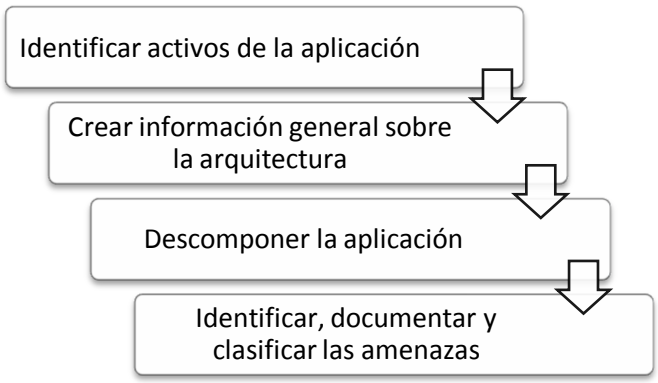


**Figura 4. Modelo de amenazas según la iniciativa *Trustworhty Computing* [38].**

La Guía del *Open Web Application Security Project* (OWASP) [61] propone un modelado de amenazas de cinco pasos basado en el *Microsoft Threat Modeling Process, como se muestra en la* figura 5.

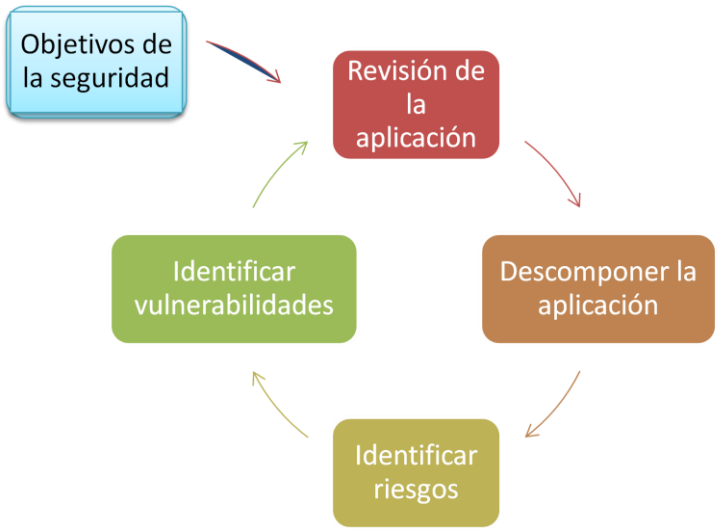


**Figura 5. Pasos del proceso de modelado de amenazas [61].**

### 1.6.2. CORAS

*CORAS* (*Consultative Objective Risk Analysis System*), es un método de ocho pasos que conduce el análisis de riesgos. Para la documentación de los resultados y las conclusiones globales se usan diagramas UML (*Unified Modeling Language*) [31; 55].

### 1.6.3. CTMM

La metodología de la empresa *Practical Threat Analisys* (*PTA*) Technologies, nombrada *CTMM (Calculative Threat Modeling Methodology)*, plantea que antes de comenzar el proceso de modelado, el analista debe familiarizarse con la aplicación. Resultando particularmente útil recopilar la siguiente documentación con el fin de que nos sirva de ayuda en el momento de decidir si se aplican o no los distintos escenarios del sistema que vamos a modelar [47]:

- Descripción funcional del sistema donde se incluyan casos de uso típicos.
- Diagrama de la arquitectura del sistema y documentación de los distintos módulos.
- Un diccionario de términos, donde se expliquen los distintos vocablos utilizados en los restantes documentos.

Esta metodología se compone de cuatro etapas: identificación de los activos, identificación de las vulnerabilidades, definición de contramedidas y creación de escenarios de amenazas y planes de mitigación.

### 1.6.4. TRIKE

Trike es un *framework* de modelado de amenazas con similitudes al proceso de modelo de amenazas de Microsoft [61].

Utiliza un método basado en riesgo, pero se diferencia porque en lugar de usar un modelo de amenaza mixto aunando ataques, amenazas y vulnerabilidades, los separa para distinguirlos unos de otros, lo que permite construir un sistema experto para la toma de decisiones.

### 1.6.5. CVSS

*CVSS (Common Vulnerability Scoring System)* no es un modelo en sí, sino que permite normalizar las notificaciones de seguridad asignándole una métrica única a las amenazas descubiertas [41].

### 1.6.6. Pasos más comunes del modelado de amenazas

Después de un análisis varios modelos existentes, se puede decir que básicamente un modelado de amenazas debería ser capaz de [31; 35; 37; 38; 47; 55; 61]:

- Identificar amenazas potenciales así como las condiciones necesarias para que un ataque se logre llevar a cabo con éxito.
- Facilitar la identificación de las condiciones o aquellas vulnerabilidades que, una vez eliminadas o contrarrestadas, afectan a la existencia de múltiples amenazas.
- Proporcionar información relevante sobre cuáles serían las contramedidas más eficaces para contrarrestar un posible ataque y/o mitigar los efectos de la presencia de una vulnerabilidad en nuestro sistema/aplicación.
- Proveer información sobre cómo las medidas actuales (tomadas hasta el momento del análisis) previenen la consecución de ataques.
- Transmitir a la gerencia la importancia de los riesgos tecnológicos en términos de impacto de negocio.
- Proporcionar una estrategia sólida para evitar posibles brechas de seguridad.
- Facilitar la comunicación y promover una mejor concienciación sobre la importancia de la seguridad.
- Simplificar la posterior actualización mediante el uso de componentes reutilizables.

Del análisis también se extrae que los pasos más comunes a seguir, para la creación del modelado son [31; 35; 37; 38; 47; 55; 61]:

1º. Crear un equipo de trabajo
2º. Descomponer la aplicación
3º. Determinar las amenazas al sistema
4º. Organizar las amenazas en orden descendente del riesgo
5º. Seleccionar la forma de responder a las amenazas

6º. Seleccionar las técnicas para mitigar las amenazas

7º. Seleccionar las tecnologías apropiadas para las técnicas identificadas

#### *1.6.6.1. Conformar un grupo de análisis de riesgos.*

El equipo de análisis de riesgos debería componerse de especialistas en diferentes tareas del proceso de desarrollo, ejemplo: Analistas de negocio, Desarrolladores, Probadores.

Aquí surge un elemento importante a la hora de querer incluir esta técnica en un ciclo de vida del s*oftware*: la necesidad de contar o no con expertos de seguridad en el grupo de analistas. La metodología puede implementarse en ambas situaciones, o en casos intermedios. Poder encarar el modelado utilizando recursos humanos no especializados en seguridad puede ser un facilitador para que el modelo sea adoptado en empresas de desarrollo con escasa o nula experiencia en esta materia [35; 37; 38; 61].

#### *1.6.6.2. Descomponer la aplicación*

Esta etapa se concentra en reconocer los componentes principales de la aplicación, incluyendo en este análisis las redes subyacentes y la infraestructura de diseño de *host*, para crear un 'perfil de seguridad' (vulnerabilidades conocidas). Esta tarea puede llevarse adelante aplicando distintas técnicas de descomposición: diagramas de flujo de datos (*DFDs*), diagramas de secuencia que permitan crear una visión general de la arquitectura, casos de uso, diagramas y tablas simples que documenten la arquitectura, incluyendo subsistemas, límites de seguridad y flujos de datos [35].

#### *1.6.6.3. Determinar las amenazas al sistema*

La lista de amenazas que se vaya conformando debe contener algunos datos como el objetivo de la amenaza, técnica de ataques en la que se puede emplear y forma de mitigar dicha amenaza [35; 37; 38; 61].

Para facilitar la determinación de las amenazas, es conveniente agruparlas en categorías, para ello se analizarán las propuestas de WASC y el STRIDE.

El Web *Application Security Consortium (WASC)* es un grupo internacional de ingenieros expertos, y organizaciones representativas que producen código abierto y que están de acuerdo con los estándares de buenas prácticas de seguridad para el

*World Wide Web*. El *Web Security Threat Classification* es una fuerza cooperativa para organizar las amenazas a la seguridad de un sitio Web, y los miembros de *WASC* han creado este proyecto para promover y desarrollar la industria de la terminología estándar para describir estos problemas. Los desarrolladores de aplicaciones, profesionales de la seguridad, comercializadores de *software* y auditores podrán contar con un lenguaje sólido para tratar los problemas relacionados con la seguridad Web [64].

Las metas del *Web Security Threat Classification* son:

- Identificar todos los tipos de ataques conocidos a la seguridad de las aplicaciones Web
- Acordar un nombre para cada tipo de ataque
- Desarrollar una estructura para organizar las clases de ataques
- Desarrollar documentación que provea una descripción genérica de cada clase de ataque

La *Web Security Threat Classification* agrupa las amenazas en seis clases y les asigna un nombre estándar para identificarlas. En la documentación se incluye una pequeña descripción de los puntos claves y ejemplos de cada amenaza [64].

No se considera ni una metodología, ni una guía o manual de seguridad, sencillamente se trata de un modelo que propone una forma de clasificar las amenazas con el fin de promover el uso de una terminología estándar.

El modelo STRIDE es otro intento similar a lo que plantea Web Security Threat Classification, que intenta categorizar las amenazas de seguridad potenciales en seis categorías, de cuyas iniciales en inglés viene el nombre STRIDE [39; 52]:

- *Spoofing Identity* (Suplantación de identidad)
- *Tampering with data* (Sabotaje)
- *Repudiation* (Repudio)
- *Information disclosure* (Revelación de información)
- *Denial of service (DoS)* (Denegación de servicio)
- *Elevation of privilege* (Elevación de privilegios)

Para seguir el método STRIDE, hay que descomponer el sistema en componentes significativos, analizar cada componente para comprobar si es susceptible a amenazas y mitigar las amenazas. A continuación, se repite el proceso hasta llegar a una situación cómoda con las amenazas restantes [52].

#### *1.6.6.4. Organizar las amenazas en orden descendente del riesgo*

Si ya se han determinado las amenazas es importante poder calcular el riesgo que estas implican para poder priorizar el trabajo partiendo de los riesgos más significativos [3].

En los análisis de riesgos cualitativos, las métricas asociadas con el impacto causado por la materialización de las amenazas se valoran en términos subjetivos, son asignadas por el analista basándose en su conocimiento y criterio. Las consecuencias de la materialización de amenazas se asocian a un determinado nivel de impacto en función de multitud de factores (pérdidas económicas efectivas, pérdida de competitividad, interrupción de negocio y pérdida de imagen). No intenta aplicar valores financieros puros a los activos, pérdidas previstas y costo de controles, sino que calcula valores relativos. Trabaja sobre la base de diferentes escenarios de riesgo y entrega un ranking de la severidad de las amenazas y la sensibilidad de los activos [14].

En los análisis de riesgos cuantitativos, las métricas asociadas con el impacto causado por la materialización de las amenazas se valoran en cifras concretas de forma objetiva. Un modelo cuantitativo habitual es aquel en el que las consecuencias de la materialización de amenazas se asocian a un determinado nivel de impacto en función de la estimación del costo económico que suponen para la organización. Sus métricas se basan en fórmulas complejas que asignan valores porcentuales o económicos. Se intenta asignar valores al costo de las medidas de control, a las pérdidas cuando se concreta una amenaza y a la probabilidad de ocurrencia de una amenaza [14].

Una forma simple de calcular el riesgo es multiplicando el valor del daño que provocaría la amenaza por la probabilidad de que esta ocurra:

$$\boldsymbol{Riesgo = DAÑO * PROBABILIDAD\ DE\ OCURRENCIA} \quad (1)$$

Donde:

***DAÑO***: Valor del daño potencial

***PROBABILIDAD DE OCURRENCIA***: Probabilidad de materialización de la amenaza

Ambos valores van desde 1 hasta 10, siendo 1 el menor valor y 10 el valor más alto en los dos casos. En el caso de la probabilidad de ocurrencia, 1 representa una amenaza lejana a ocurrir, mientras 10 representa una amenaza cercana, de manera similar, para el valor del daño potencial 1 indica el mínimo daño y 10 representa una catástrofe. Utilizando este acercamiento, el riesgo presentado por una amenaza con probabilidad baja, pero con alto daño potencial, es igual al riesgo planteado por una amenaza de bajo daño potencial y alta probabilidad de ocurrencia [14].

Existen métodos reconocidos para la evaluación de las amenazas y cálculo del riesgo, entre los más conocidos están: DREAD, *ACE Team y Octave.*

DREAD es un modelo obtenido del trabajo de Microsoft y empleado para evaluar la amenaza. Su nombre proviene del acrónimo de los cinco aspectos fundamentales que son utilizados para medir cada vulnerabilidad [40]:

**D**año Potencial: ¿Cuán grande será el daño si se explota la vulnerabilidad?
Facilidad de **R**eproducción: ¿Cuán fácil será reproducir el ataque?
Capacidad de **E**xplotación: ¿Cuán fácil será iniciar un ataque?
Usuarios **A**fectados: ¿Qué por ciento de usuarios será afectado?
Dificultad para su **D**escubrimiento: ¿Cuán fácil es encontrar la vulnerabilidad?

Aquí propone calcular el riesgo igualándolo al promedio de los valores de los cinco aspectos [5]:

$$\boldsymbol{Riesgo} = ((\boldsymbol{DAÑO} + \boldsymbol{FACILIDAD\ DE\ REPRODUCCIÓN} + \boldsymbol{CAPACIDAD\ DE\ EXPLOTACIÓN} + \boldsymbol{USUARIOS\ AFECTADOS} + \boldsymbol{FACILIDAD\ DE\ DESCUBRIMIENTO})/\mathbf{5}$$

Este modelo tiene un carácter subjetivo, pues se hace un tanto difícil trabajar con él cuando se tiene varios especialistas que deben ponerse de acuerdo y decidir qué evaluación otorgar a cada uno de estos aspectos frente a cada amenaza particular,

además de que se trabaja sobre la base de la estimación, en la que la experiencia del personal tiene especial importancia.

*ACE Threat Analysis and Modeling (ACE Team)* es resultado de la revisión de DREAD, en la que la perspectiva de análisis se sitúa desde el punto de vista del defensor [5]. Este modelo centra el proceso en una idea: un evento no es una amenaza si no se traduce en un impacto negativo para el negocio; pero se mantiene la idea original de realizar el análisis y modelado como un proceso iterativo desde las etapas tempranas del desarrollo del *software*, enriqueciéndolo a medida que se avanza por las etapas y evoluciona la aplicación.

OCTAVE *(Operationally Critical Threat, Asset, and Vulnerability Evaluation)* ordena los recursos críticos, las amenazas y vulnerabilidades asociadas a esos recursos y los niveles de riesgo. El usuario es guiado a través de un proceso paso a paso para catalogar los riesgos y generar los "Perfiles de amenazas" a través de los diagramas de flujo que traza un recurso crítico hacia sus fuentes de riesgo potencial [41; 46].

### 1.6.7. Seleccionar la forma de responder y mitigar las amenazas

Se tienen cuatro formas de responder a las amenazas: No hacer nada; Informar al usuario; Eliminar el problema; Remediar el problema.

En dependencia de la herramienta utilizada para determinar las amenazas del sistema, esta puede sugerir la técnica para mitigarlas. En el caso de STRIDE, este ofrece información sobre las técnicas necesarias [35].

### 1.6.8. Seleccionar las tecnologías apropiadas para las técnicas identificadas

Técnicas no es lo mismo que tecnologías, por ejemplo, Autenticación es una técnica de seguridad y existen diferentes tecnologías para implementar la autenticación. Si se tiene una lista de las amenazas y las técnicas para mitigarlas, bastaría con añadir una correspondencia con las tecnologías disponibles [35].

La situación ideal es mitigar una amenaza con una solución sólida y comprensible. Por ejemplo, se cree que el uso adecuado de criptografía sólida constituye una medida eficaz contra numerosos tipos de amenazas de revelación de información. Puede que jamás consiga probar que una defensa es perfecta. La elección de una tecnología no siempre es fácil [52].

El modelado de amenazas es una herramienta útil para la evaluación de los riesgos y selección de los controles a implementar, por lo que se tendrá en cuenta para la propuesta.

## Conclusiones parciales

Del análisis realizado hasta el momento, se puede concluir que existen técnicas, buenas prácticas y algunas guías y proyectos que intentan estandarizar el tratamiento de la seguridad en el desarrollo del software, pero son generalizadoras, necesitan ser adaptadas al entorno donde se quiere aplicar y concretar en más detalles para ser empleadas por desarrolladores de poca experiencia y no son soluciones ágiles.

El desarrollo de la seguridad es un proceso constante a lo largo de la vida útil de la aplicación y el diseño inicial influye mucho en la evolución futura de la aplicación. Por lo que en la definición de las tareas por etapas para la propuesta, se hará énfasis en tareas de evaluación para las primeras etapas y deberá contener algunas tareas repetitivas y constantes como el análisis de riesgos y la documentación de la seguridad.

# CAPÍTULO 2. SOLUCIÓN PROPUESTA

## Introducción

Partiendo de los resultados del análisis del entorno y con el objetivo de encontrar una solución a la necesidad de desarrollar software garantizando la seguridad de los datos y del propio sistema, en el presente capítulo se propone una guía metodológica para el desarrollo de la seguridad en aplicaciones informáticas.

Esta guía constituye un modelo de trabajo general, sencillo y flexible que permite su ejecución paralelamente al ciclo de vida de desarrollo del software, integrándose a este a través de algunos puntos de contactos en cada fase. Además, incluye un grupo de plantillas para apoyar a conformar la documentación que se genera con cada una de las salidas de los pasos que se realizan y así obtener un expediente de seguridad donde se describen los controles que se implantarán, las decisiones y los análisis realizados.

## 2.1.Definición de responsabilidades

Se definieron seis responsabilidades que tienen asociadas funciones a realizar para el desarrollo de la seguridad, estas responsabilidades no tienen que coincidir con los roles específicos de la metodología de desarrollo de software que se está empleando. Tampoco se requiere la inclusión de miembros nuevos al equipo de proyecto para atender la seguridad, o sea, estas funciones serán cubiertas por los miembros ya existentes.

Las responsabilidades definidas son:

1) Cliente
2) Coordinador: Es quien planea la tareas y supervisa su cumplimiento así como el estado del proyecto.
3) Analista (Responsable del equipo de desarrolladores): El analista traduce las necesidades y metas de seguridad en escenarios (casos de uso y casos de abuso) que el equipo de desarrolladores empleará para construir los controles de seguridad.

4) Arquitecto: Traduce los requerimientos a especificaciones de tecnología de información y define los elementos tecnológicos que ayuden a implantar las recomendaciones de las evaluaciones de riesgos realizadas; además, analiza el impacto tecnológico y económico de la implantación de mecanismos de seguridad.
5) Desarrollador: Escribe las pruebas unitarias y produce el código de los controles.
6) Encargado de pruebas: Ejecuta las pruebas regularmente y difunde los resultados en el equipo.

El ciclo de vida de desarrollo del software (SDLC) es un proceso global de desarrollo e implementación a través de múltiples pasos. Existen diferentes modelos y metodologías, pero básicamente todas consisten en una serie de fases definidas. Para cualquier caso, la seguridad debe integrarse dentro del SDLC con el fin de preservar el correcto funcionamiento del sistema y la información que este manipula.

## 2.2.Propuesta de guía metodológica

Se puede definir método, como el modo ordenado de proceder para llegar a un resultado o fin determinado. Se define una metodología como el conjunto de métodos por los cuales se regirá un proceso [27].

Una guía (en el contexto de este trabajo) es un documento que contiene indicaciones, preferentemente basadas en metodologías o procedimientos. Dicho documento también tendrá listas de datos o información referente a una materia y su fin es dirigir y encaminar hacia el logro de un objetivo.

En el Capítulo 1 se concluyó que no existe una solución ágil para el desarrollo de la seguridad, que sea práctica y adaptable a las características de los proyectos que se desarrollan en la UCI, muy limitados en tiempo y con carencia de personal especializado en temas de seguridad.

La guía metodológica que se propone se basa en los principios del manifiesto ágil y en el estándar *ISO/IEC 12207:1995* de la *IEEE* [18]. Las cinco etapas que la componen, representadas en la Figura 6, son una variante fundamentada en los cinco procesos primarios definidos en dicho estándar y en las actividades principales de una estrategia de manejo de riesgos [43].


**Figura 6. Etapas de la Guía Metodológica para la implementación de la seguridad durante el desarrollo de aplicaciones informáticas.**

Para la definición de las tareas y salidas de cada una de ellas se tuvo en cuenta el análisis realizado, en los epígrafes 1.4 y 1.5 del Capítulo 1, sobre los siete puntos de contacto para la seguridad del software [15] y los principios de seguridad propuestos por varios autores [21; 35; 36].

La primera tarea fundamental corresponde a la gestión de requisitos, la cual posibilita la comunicación entre el cliente y el equipo de desarrolladores. De esta tarea surge la especificación preliminar de los requisitos que constituyen la base primaria para el análisis y diseño de la arquitectura.

En la figura 7, se muestra una vista abstracta de la seguridad a la que se le ha aplicado un modelo de procesos V.

El modelado de amenazas se emplea como una técnica a la que se recurre para hacer una revisión del diseño o de la arquitectura del sistema a fin de encontrar posibles problemas de seguridad en el diseño y poder corregirlos. Se nutre de los requerimientos de seguridad y de la arquitectura del sistema para determinar las amenazas y vulnerabilidades.

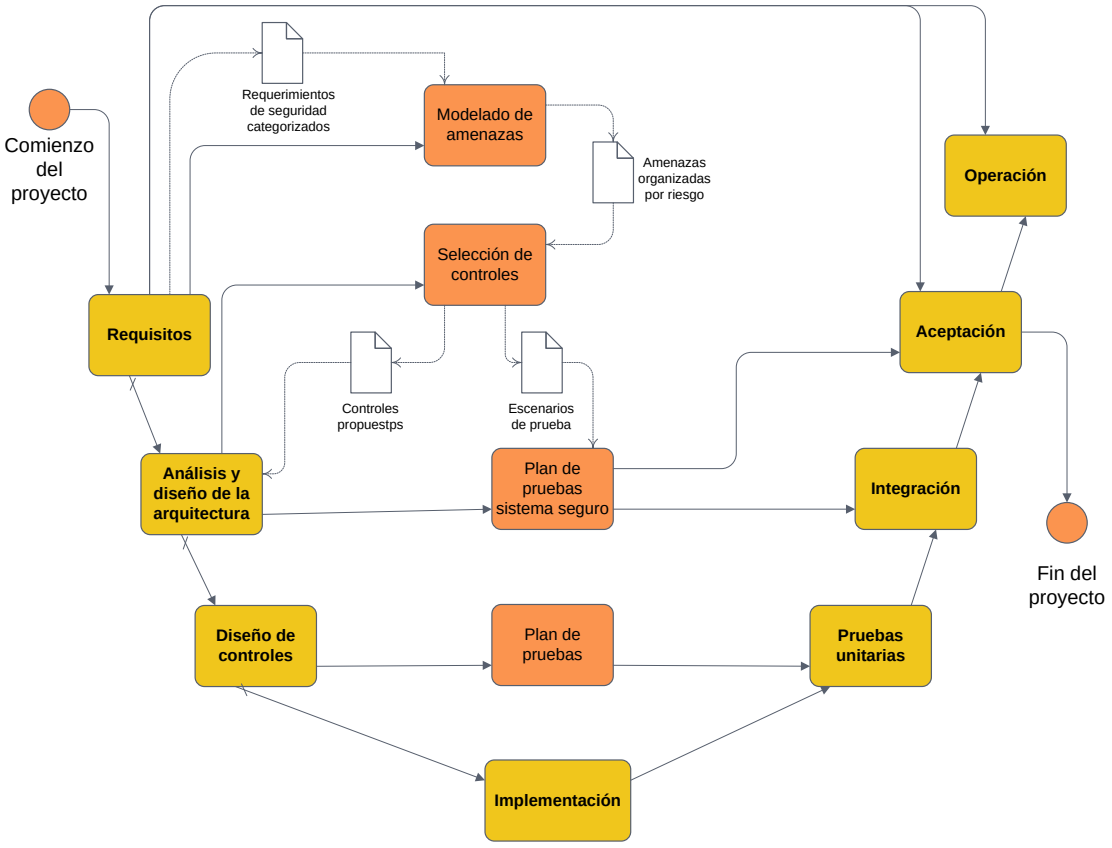


**Figura 7. Modelo conceptual de la guía metodológica propuesta**

Como resultado del análisis de riesgo de las amenazas detectadas, se obtienen los controles que formarán parte de la arquitectura de seguridad y los escenarios de prueba.

Con los escenarios de prueba se conforma el plan de pruebas del sistema. El plan de pruebas se llevará a cabo una vez que se realice la integración de los controles de seguridad, al sistema desarrollado.

### 2.2.1. Etapa "Inicio y planificación"

En esta etapa se asignan las responsabilidades de seguridad para el desarrollo del sistema, teniendo en cuenta que en un desarrollo ágil no se introducen nuevos miembros al equipo para atender la seguridad. El coordinador es el que lleva la mayor carga durante esta etapa, aunque todos participan.

Este es el momento en que se identifican las necesidades del sistema y se categorizan los requerimientos para luego ser usados junto a las amenazas y vulnerabilidades para

analizar el riesgo. Se normalizan los procesos que consideren prácticas seguras y sean apropiados para el nivel de seguridad requerido y se identifican las fuentes de requerimientos de seguridad, como leyes, regulaciones y estándares. Esta estapa se compone de cinco pasos como muestra la figura 8.

El desarrollo de un perfil de proyecto inicial para los hitos de seguridad que se integren dentro del cronograma de desarrollo del proyecto, permitirá una planificación adecuada cuando se produzcan cambios. El equipo de desarrollo (incluyendo a los encargados de prueba) necesita entrenamiento en temas de seguridad para llevar a cabo el diseño e implementación de los controles, por lo que en esta etapa se planifica un plan de entrenamiento.

Independientemente de que la mayor parte de las decisiones de planificación se realizan durante la ejecución de este paso, es posible planear hitos o reuniones a lo largo del desarrollo del sistema de seguridad para discutir algunas consideraciones.

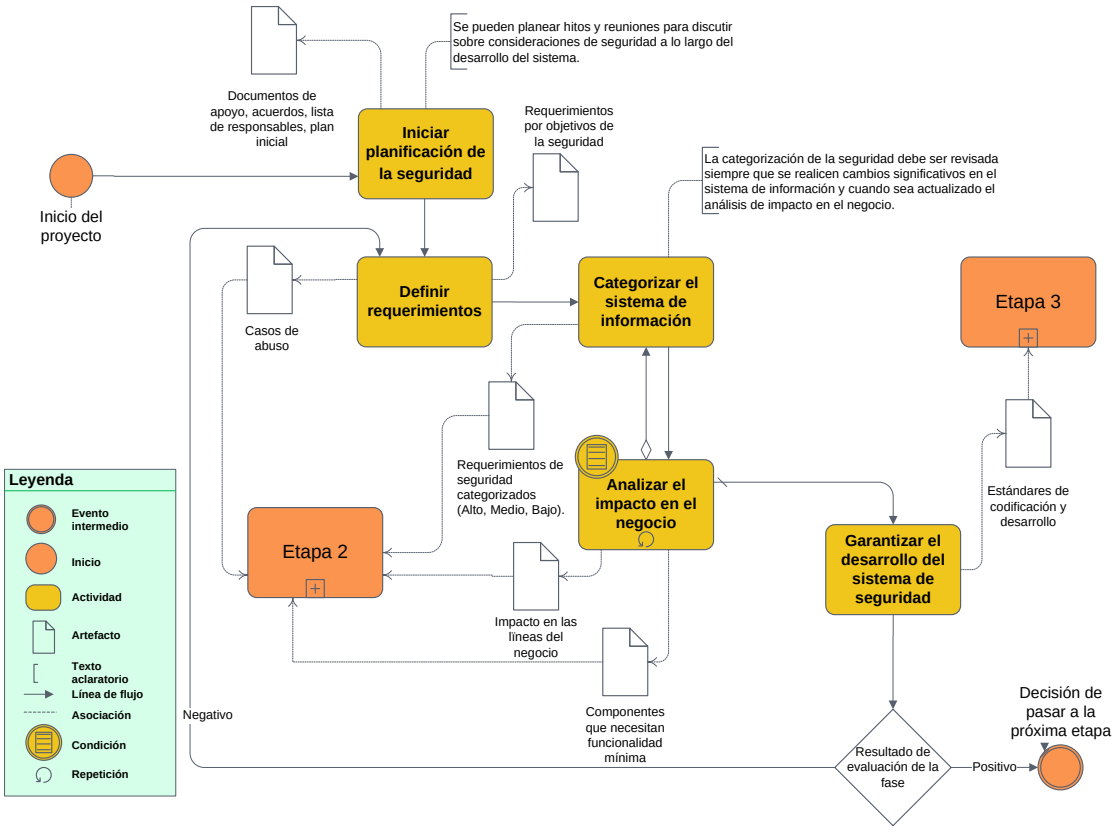


**Figura 8. Etapa 1: Inicio y planificación**

Para la implementación de esta tarea deben tenerse presente los siguientes puntos:

La planificación de la seguridad en la fase inicial debe incluir los preparativos para el ciclo de vida completo, incluyendo la identificación de los hitos clave de seguridad y los resultados; herramientas y tecnologías. Se debe prestar especial atención a los elementos que pueden necesitar ser adquiridos, ejemplo: las herramientas de prueba y evaluación.

El cronograma general del proyecto debe integrar las actividades de seguridad para garantizar una adecuada planificación de futuras decisiones relacionadas con la planificación y los recursos.

#### *2.2.1.1. Definir requerimientos*

En este paso es cuando se identifican las necesidades del sistema y se documentan. La responsabilidad fundamental en este paso es la de analista.

Para la definición de los requerimientos se identificaron dos niveles, que son los que esta guía propone (ver la tabla 2); pero se puede definir otro patrón de categorización de acuerdo a las necesidades o características del sistema.

Las salidas de este paso son los objetivos de la seguridad y los requerimientos del negocio ordenados en términos de confidencialidad, integridad y disponibilidad, según el objetivo de seguridad al que responda. También es salida de este paso la especificación de los casos de abuso que tendrán utilidad en la Etapa 2.

**Tabla 2. Niveles de categorización de los requerimientos**

| I Nivel de categorización | II Nivel de categorización |
|---|---|
| No repudio | No repudio |
| Integridad | Modificación<br>Borrado<br>Validación de datos<br>Manipulación de excepciones<br>Separación de responsabilidades |

| | |
|---|---|
| Disponibilidad | Tiempo de respuesta<br>Expiración<br>Asignación de recursos |
| Confidencialidad | Cifrado<br>Autenticación<br>Aprobación<br>Trazas |

#### *2.2.1.2. Categorización del sistema de información*

Para evaluar el riesgo posteriormente, se necesita la categorización de la información que el sistema maneja. En este paso intervienen las responsabilidades del coordinador y el analista, pero se debe involucrar al cliente, debido a que es quien tiene mayor claridad en cuanto a la naturaleza de la información que se quiere categorizar.

La categorización de la información debe ser revisada cuando sea actualizado el análisis de impacto en el negocio y se hará siguiendo lo dispuesto en la ley vigente (Decreto Ley 199/99 Sobre la Seguridad y Protección de la Información Oficial).

De los parámetros principales que pueden ayudar en la categorización están:

- Alcance de la aplicación: intranet o Internet
- Sensibilidad de los datos
- Tecnología usada: basado en web o no basado en web

#### *2.2.1.3. Evaluar el impacto en el negocio*

La valoración de impacto en el negocio es utilizado para caracterizar las consecuencias de la ruptura de cada componente del sistema. A medida que el sistema madure debe aumentarse este análisis con más detalles, las responsabilidades que intervienen son las relacionadas con analista y cliente.

La salida principal de esta actividad es a la identificación del impacto, ante una eventualidad, de las líneas del negocio apoyadas por este sistema. Lo cual será de mucha utilidad durante el análisis de riesgo, donde se necesita evaluar que riesgos se van a asumir y cuáles deben ser tratados con más o menor prioridad.

La información de la valoración del impacto en el negocio es empleada para determinar los requerimientos de los procesos ante contingencias y las soluciones de mitigación.

Para la implementación de esta tarea deben tenerse en cuenta las siguientes consideraciones:

- La presencia de stakeholders en reuniones y tormentas de ideas para debatir elementos relativos a esta actividad.
- Debe repasarse periódicamente y ponerse al día cuando se tomen decisiones de desarrollo o cuando ocurran cambios significativos en el alcance y propósito del sistema.

#### *2.2.1.4. Garantizar el desarrollo del sistema de seguridad*

El desarrollo del sistema se debe realizar con procesos normalizados que consideren prácticas seguras y sean apropiados para el nivel de seguridad requerido; además, deberán quedar documentados. El equipo de desarrollo, incluyendo los probadores, necesita entrenamiento en temas de seguridad para llevar a cabo el diseño e implementación de los controles.

El ambiente de desarrollo tiene que ser seguro, eso incluye las estaciones de trabajo, los servidores, dispositivos de red y repositorios de código. Ejemplo, los accesos a los repositorios de código deben ser restringidos.

Este paso es responsabilidad fundamental del coordinador, pero el analista interviene también.

Se tiene como salidas los estándares de codificación y desarrollo, incluyendo el ambiente de desarrollo y el plan de entrenamiento del equipo en temas de seguridad, lo cual es necesario a fin de incrementar el entendimiento de los temas y técnicas necesarias para desarrollar sistemas seguros.

Buscar y hacer pruebas durante la etapa de desarrollo disminuye el número de defectos de seguridad, si los probadores están bien entrenados podrán identificar más fácilmente problemas de seguridad antes que el producto pase a la siguiente etapa.

#### *2.2.1.5. Evaluación de la etapa*

Para la evaluación de la etapa se realizan las siguientes tareas:

- Análisis de la idea del sistema que verifique que esta es viable, cumplible y está acorde con los objetivos de la organización y las restricciones de presupuesto.
- Análisis de las especificaciones de ejecución que asegure que el proyecto inicial del sistema está direccionado hacia todos los requisitos de seguridad especificados.
- Análisis de los resultados de la categorización de seguridad del sistema de información.

### 2.2.2. Etapa "Análisis y diseño de la arquitectura de seguridad"

En esta etapa, antes de la aprobación de especificaciones planeadas se evalúa el riesgo para el sistema, ya que se pueden generar necesidades de especificaciones adicionales; también se hace con el objetivo de evaluar cómo el sistema podría afectar otros sistemas directa o indirectamente.

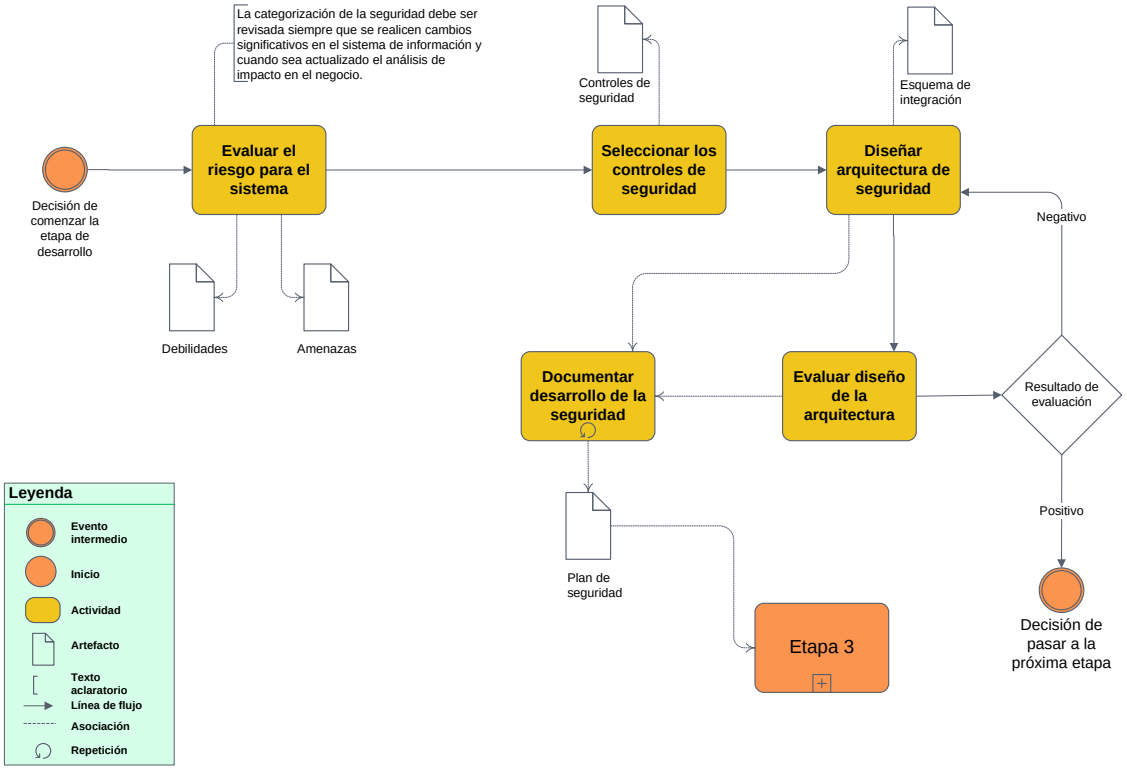


**Figura 9. Etapa 2: Análisis y diseño de la arquitectura de seguridad.**

Los controles a desarrollar son cuidadosamente seleccionados y el diseño de la arquitectura de seguridad tiene que tomar en consideración los servicios externos y las

interconexiones de sistemas; así como la estrategia de auditoría del sistema, que debe ser desarrollada para habilitar un análisis de trazas preciso o reconstruir todos los flujos de trabajo de alto riesgo y prioridad.

A partir del resultado de evaluación del riesgo los requerimientos deben ser analizados nuevamente.

Esta etapa está compuesta por seis tareas como se muestra en la figura 9, que serán ejecutadas fundamentalmente por los encargados de cumplir las responsabilidades de analista y arquitecto.

#### *2.2.2.1. Evaluar el riesgo para el sistema*

La valoración de riesgo de seguridad debe dirigirse antes de la aprobación de especificaciones planeadas ya que se pueden generar necesidades de especificaciones adicionales.

También se hace con el objetivo de evaluar cómo el sistema podría afectar otros sistemas al que será directa o indirectamente conectado.

Las salidas de este paso se resumen en el documento generado como resultado del modelado de amenazas, que entre otros detalles, contendrá la lista de amenazas organizadas en orden descendiente del riesgo y las vulnerabilidades del sistema.

El cálculo del riesgo se propone realizarlo empleando un método híbrido, tomando la flexibilidad que ofrecen los métodos cualitativos y aprovechando la precisión de los métodos cuantitativos. Según el resultado del análisis realizado en el Capítulo 1, la mejor elección como método cuantitativo es DREAD, sobre todo por su simplicidad y flexibilidad. La guía ofrece un manual detallado con el procedimiento para realizar el modelado de amenazas y algunas sugerencias de métodos para realizar los cálculos y estimaciones necesarias. No obstante, el equipo de desarrollo puede tomar otro método.

La valoración de riesgo se completa en una fase más madura del desarrollo del sistema y puede haber una necesidad de volver a visitar las actividades previas, como la categorización de seguridad por algún cambio de requisitos.

#### *2.2.2.2. Seleccionar los controles de seguridad*

Los controles de seguridad son una definición abstracta de lo que posteriormente serán los módulos, componentes o sistemas que se implementarán para mitigar los riesgos de seguridad.

La selección de los controles debe hacerse teniendo en cuenta el principio de la proporcionalidad, que plantea destinar los recursos necesarios para implementar un control siendo consecuentes con lo que se desea proteger. Para esto, los controles de seguridad se seleccionan a partir del resultado del análisis de riesgos y se basa en tres actividades fundamentales:

- Selección de líneas base de seguridad
- Valorar posibles controles de seguridad ajustándolos a la línea base inicial
- Adición de suplementos a la línea base, constituidos por controles basados en la valoración de riesgo y las condiciones locales.

La línea base es la definición de hasta donde se quiere llegar y se está dispuesto a tolerar.

Este paso tiene como única salida la lista de controles, que se documentan en el plan de seguridad del sistema junto a todas las decisiones tomadas para la selección.

#### *2.2.2.3. Diseñar arquitectura de seguridad*

El diseño de la seguridad a nivel de sistema tiene que tomar en consideración, los servicios externos y las interconexiones entre sistemas. También debe tener en cuenta la estrategia de auditoría del sistema que deberá ser desarrollada para habilitar un análisis de trazas preciso o reconstruir todos los flujos de trabajo de alto riesgo y prioridad.

Las salidas de este paso son, el esquema detallado de integración de seguridad, la lista de servicios compartidos y riesgos compartidos resultantes, así como la identificación de controles comunes usados por el sistema.

Todos los detalles respecto al diseño de la arquitectura se documentan en el plan de seguridad del sistema.

#### *2.2.2.4. Evaluación de la etapa*

Para la evaluación de la etapa se realizan las siguientes tareas:

- Análisis del diseño y la arquitectura que evalúe el diseño del sistema planificado y la posible integración con otros sistemas, así como incorporación de servicios compartidos y controles de seguridad comunes (autenticación, recuperación de desastre, detección de intrusos o reporte de incidentes)
- Análisis del rendimiento del sistema que evalúe si el sistema da o es capaz de dar las expectativas documentadas del cliente y si el sistema se comporta de una manera predecible si es sometido a un uso incorrecto.
- Análisis de las funciones del sistema que garanticen que los requerimientos funcionales identificados están suficientemente detallados y pueden ser probados.
- Análisis de las decisiones en la gestión de riegos.

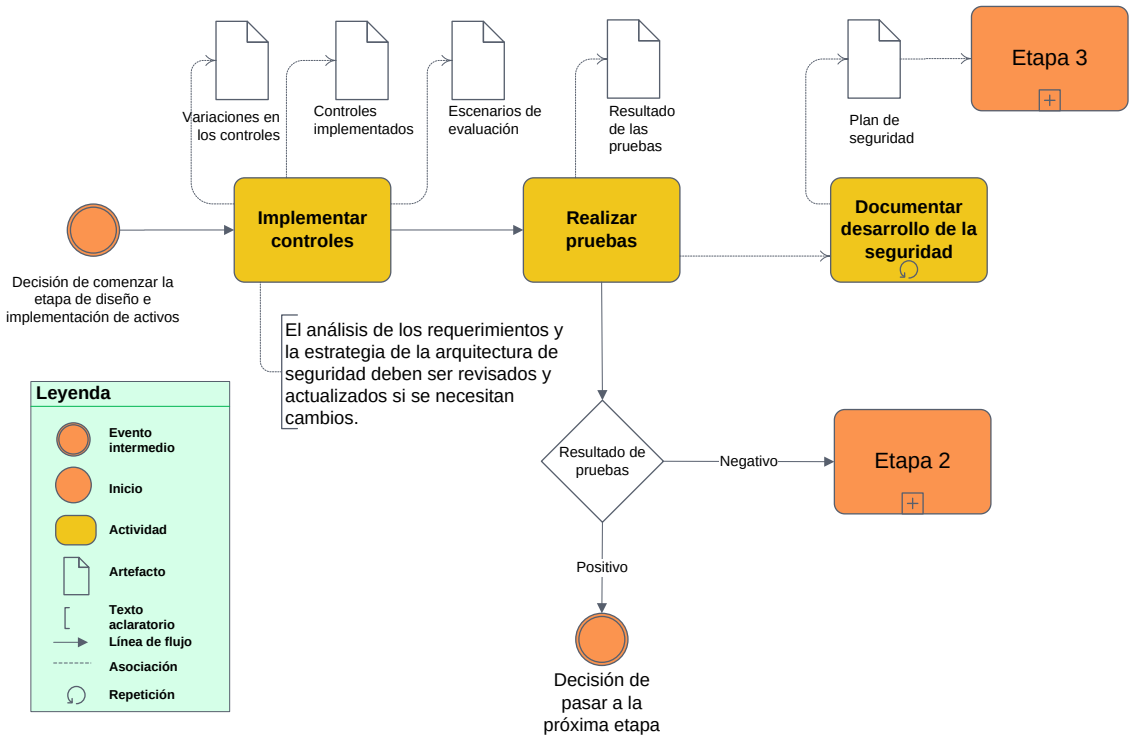


**Figura 10. Etapa 3: Implementación de controles**

### 2.2.3. Etapa "Implementación de controles"

En esta etapa los controles de seguridad son implementados y convertidos en parte del sistema en vez de aplicarlos al final. Antes de implantar el sistema de seguridad deben realizarse un conjunto de pruebas a los controles desarrollados y todas las decisiones tomadas y consideraciones, deben quedar documentadas. En la figura 10 se muestran las tareas para esta etapa, que son ejecutadas por quienes cumplen funciones de desarrollador.

#### *2.2.3.1. Implementar controles*

Las salidas son la lista de controles implementados y las variaciones realizadas a los mismos según lo planificado, así como, los escenarios de evaluación para probar vulnerabilidades conocidas o limitaciones.

Al realizar esta actividad deben cumplirse rigurosamente las disposiciones tomadas para garantizar un ambiente de desarrollo seguro y tener en cuenta los estándares de codificación y desarrollo definidos en la Etapa "Inicio y Planificación".

#### *2.2.3.2. Realizar pruebas*

Antes de implantar el sistema de seguridad deben realizarse un conjunto de pruebas a los controles implementados, puede involucrarse al Encargado de pruebas o pueden llevarse a cabo por los mismos desarrolladores.

Podrán seleccionarse las herramientas para realizar pruebas de acuerdo a la tecnología empleada para el desarrollo del sistema.

### 2.2.4. Etapa "Integración/Prueba"

Durante esta etapa es cuando la seguridad será instalada junto al sistema, la figura 11 muestra los pasos a seguir.

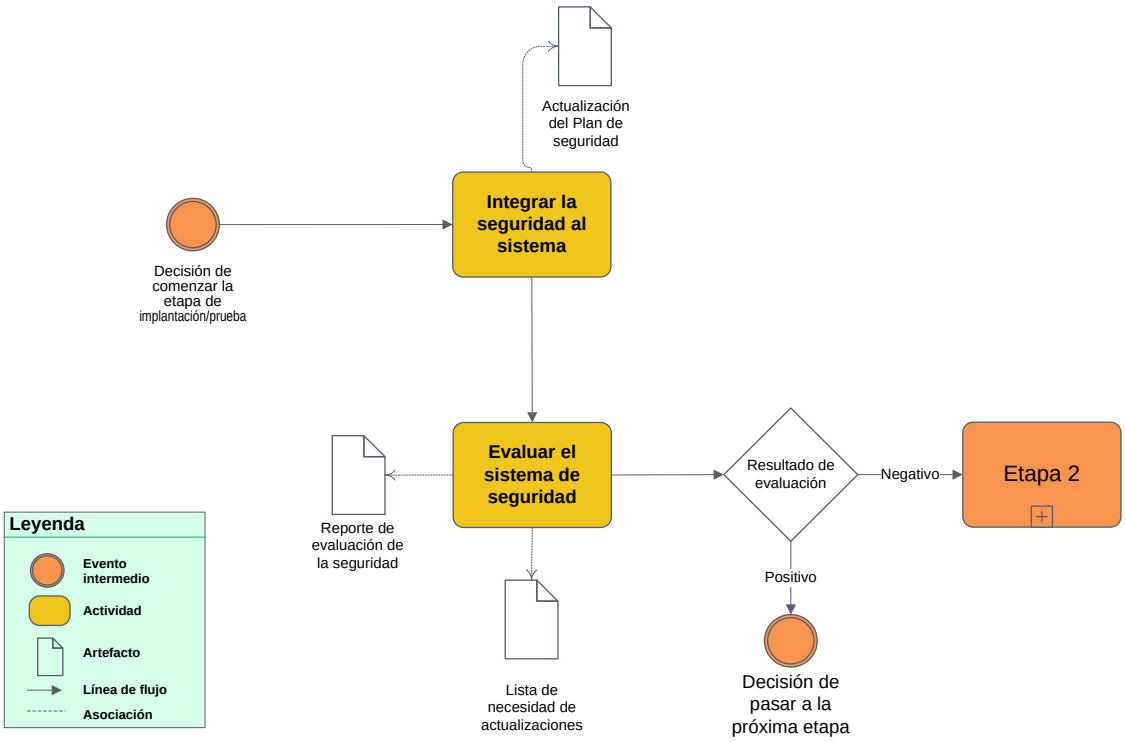


**Figura 11. Etapa 4: Integración/Prueba**

Cuando se requieran modificaciones o cambios, el sistema regresará a una etapa anterior en correspondencia con el problema detectado. El encargado de pruebas lleva el mayor peso en esta etapa junto a al desarrollador.

#### *2.2.4.1. Integrar la seguridad al sistema*

Este paso requiere la actualización del plan de seguridad del sistema, en caso de no ser el primer intento de integración, pues el sistema puede haber regresado a alguna etapa previa debido a las necesidades de actualización generadas al efectuar la evaluación.

#### *2.2.4.2. Evaluar el sistema de seguridad*

El objetivo de las pruebas y el proceso de evaluación es validar que el sistema desarrollado cumple con los requisitos funcionales y de seguridad.

Las salidas de este paso son la lista de necesidades de evaluación y el reporte de evaluación del sistema de seguridad.

### 2.2.5. Etapa "Operación/Aceptación"

En esta etapa el sistema es supervisado permanentemente mediante un monitoreo continuo en su desempeño con el objetivo de determinar cómo puede ser más eficaz, seguro y eficiente el funcionamiento.

Como muestra la figura 12, si es necesario deberán realizarse cambios en la configuración. Las operaciones de mantenimiento continuarán hasta que el sistema se adapte y responda eficazmente a las necesidades de la organización, manteniendo los niveles de riesgo aceptados.

Debido a las incidencias detectadas durante el monitoreo, puede ser necesario hacer cambios de configuración o tomar medidas para controlar la causa de las incidencias. La salida del monitoreo son las incidencias detectadas, en dependencia de estas se requerirá volver al paso de configuración y control o regresar a una etapa anterior. El documento de aceptación firmado por el cliente cierra el desarrollo del sistema.

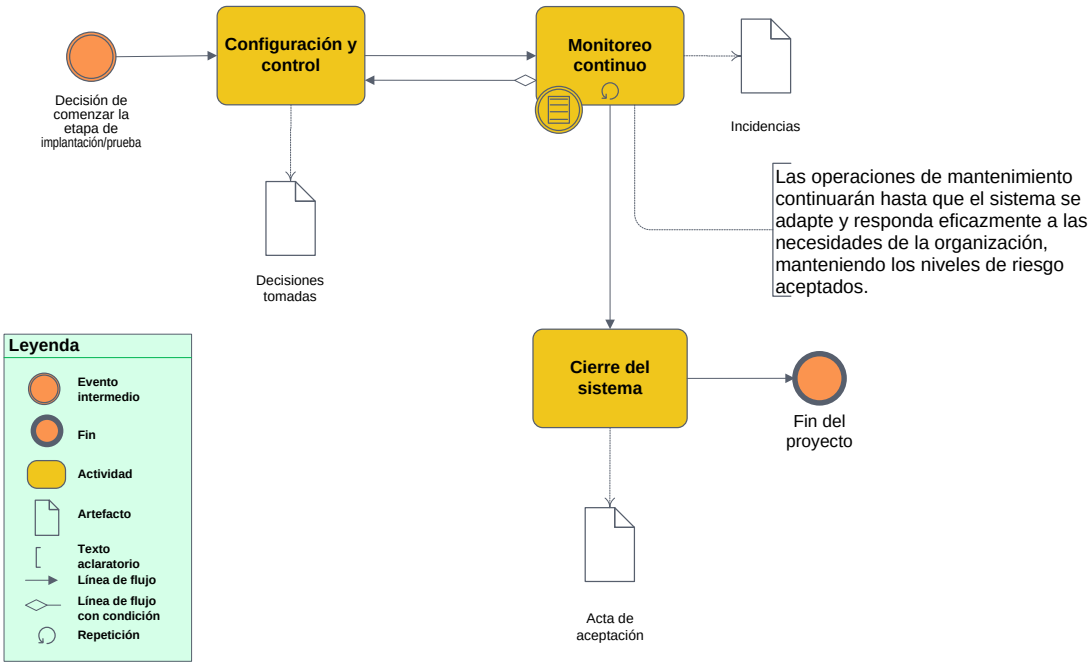


**Figura 12. Etapa 5: Operación/Aceptación**

### 2.2.6. Documentación del desarrollo de la seguridad

La documentación del desarrollo de la seguridad, generará la actualización constante del plan de seguridad del sistema. Esta Guía sugiere la utilización de una plantilla basada en las recomendaciones del *National Institute of Standards and Technology* (NIST) [43] para la confección de dicho plan.

A pesar de que sitúa la tarea de documentar como un paso de la Etapa 2, la documentación se realiza constantemente durante todo el proceso y debe servirse fundamentalmente de las salidas de las actividades:

- Definición de requerimientos
- Categorización del sistema de información
- Evaluación de impacto en el negocio
- Diseño de la arquitectura de seguridad
- Implementación de controles
- Resultados de pruebas realizadas

## 2.3.Modelado de amenazas

El modelado de amenazas se inicia desde las primeras etapas y aunque en esta propuesta se sugiere llevarlo a cabo para realizar la tarea de evaluación de riesgos, el modelado de amenazas puede realizarse en cualquier momento después de tener la arquitectura del sistema que se quiere asegurar y se haya evaluado el impacto en las líneas del negocio.

La lista de amenazas podrá ser organizada a partir de bibliotecas existentes como las de CWE (*Common Weakness Enumeration*) [57], OWASP (*Open Web Application Security Project*) [62] o CAPEC [7]. También puede ser a partir de las propuestas de WASC [64] y STRIDE [39] o alguna otra clasificación que se decida.

Esta guía no impone la utilización de un método en particular para el modelado de amenazas, ni siquiera sugiere que sea necesario el empleo de un método, solo propone una plantilla que contiene la descripción paso a paso de los elementos necesarios para llegar a la selección de los controles de seguridad. Aun así, si el equipo de desarrollo lo estima conveniente, puede emplear algún método de modelado de amenazas conocido.

## 2.4.Generalización de la solución

Para publicar la guía práctica desarrollada, es necesario seleccionar un medio que sea de fácil acceso y transportable.

Estas características fueron fácilmente encontradas en una plataforma que permite el trabajo colaborativo y la organización de la información siguiendo diferentes criterios.

La plataforma seleccionada fue *Mediawiki* [32] que es uno de los Sistemas de Contenidos más populares que se especializan en wikis y es público bajo la Licencia *GNU GPL*, está programado en *PHP* usando *MySQL* sobre *Apache*.

*Mediawiki* cuenta con las herramientas básicas para la creación de contenidos en forma colaborativa. Soporta múltiples usuarios y diferentes niveles de acceso, capacidad para manejar varios formatos de imagen.

La guía metodológica publicada sobre esta plataforma permite la organización dinámica del contenido por: salidas, roles y buenas prácticas. Permite la publicación de información adicional como métodos, listas de amenazas y herramientas que son de utilidad para la ejecución de los pasos.

## Conclusiones parciales

La Guía metodológica para implementar la seguridad durante el desarrollo de aplicaciones informáticas, fue concebida bajo los principios de desarrollo ágil. El orden de las tareas y la estructura de la Guía están fundamentados por estándares internacionales así como buenas prácticas y principios para el desarrollo de software seguro.

Cada paso propuesto en esta guía para implementar la seguridad de aplicaciones informáticas, está detallado indicando objetivo, entradas y salidas de artefactos, roles que lo ejecutan, buenas prácticas a tener en cuenta y la interacción y dependencia de otros pasos.

Siempre que se respete el objetivo y las salidas de una tarea se pueden emplear los métodos o herramientas que el equipo de desarrollo entienda. El modelado de amenazas es fundamental, ayuda en la mejor planificación para mitigar las amenazas del sistema informático o a corregir errores cometidos.

La propuesta se pone a disposición de los equipos de desarrollo de sistemas informáticos mediante una plataforma de trabajo colaborativo junto a herramientas y documentación relativa a la seguridad de aplicaciones.

# CAPÍTULO 3. EVALUACIÓN DE LA GUÍA METODOLÓGICA PROPUESTA

## Introducción

En el presente capítulo se realiza la evaluación de la guía propuesta. Para ello se utiliza el método Delphi de evaluación por criterio de experto, que permiten dar un pronóstico sobre la efectividad de la solución.

Se describen cada una de las etapas del método: elección de expertos, elaboración y lanzamiento de los cuestionarios y evaluación de los resultados.

## 3.1.Método seleccionado para evaluar la propuesta

Existen diferentes métodos para evaluar la propuesta, algunos son: el *Test* de *Turing*, el de Validación de comportamientos en casos extremos y el método *Delphi*.

Se seleccionó como método de evaluación para la presente propuesta el método *Delphi*, pues para propuestas de este tipo la aplicación y recolección de datos puede demorar mucho tiempo y en este caso, aun no se tienen suficientes datos sobre los resultados obtenidos al aplicar la propuesta.

El método *Delphi* permite obtener las opiniones de un panel de especialistas interrogando de forma individual, iniciando con una encuesta y evitando el encuentro para garantizar el anonimato e impedir la influencia de unos en otros. Este proceso se repite las veces que sean necesarias para llegar a un consenso.

## 3.2.Selección de los expertos

Para la selección de los expertos se identificaron diez especialistas y se envió, en una primera ronda de cuestionamiento (Anexo 2), un documento en el que se le solicita su conformidad con ser experto y se le plantea que responda un cuestionario de autovaloración en cuanto al dominio del tema y las fuentes de argumentación a partir de las cuales ha logrado ese conocimiento.

Se estableció el coeficiente de competencia para cada posible experto a partir de la fórmula: $K = {}^{1}/_{2}\,(k_c + k_a)$

**Donde:**

K: Coeficiente de competencia de cada experto (Ver resultados en la tabla 3).

kc: Coeficiente de conocimiento o información que tiene el experto acerca del problema. Según lo establecido en esta metodología se le pide a cada experto es que marquen en una escala creciente de 1 a 10, con una cruz, el valor que se corresponde con el grado de conocimiento o información que tienen sobre el tema de estudio. El coeficiente se calcula multiplicando la valoración del propio experto por 0,1.

ka: Este coeficiente se autoevalúa en alto (A), medio (M) o bajo (B) para un grupo de aspectos que influyen sobre el nivel de argumentación o fundamentación del tema a estudiar (Ver Anexo 2).

**Tabla 3. Resultados del procesamiento para la determinación del coeficiente de competencia**

| Experto | $K_c$ | $K_a$ | K | Valoración |
|---|---|---|---|---|
| 1 | 0.6 | 0.7 | 0.65 | M |
| 2 | 0.2 | 0.8 | 0.5 | M |
| 3 | 0.6 | 0.9 | 0.75 | M |
| 4 | 0.7 | 0.6 | 0.65 | M |
| 5 | 0.6 | 0.8 | 0.7 | M |
| 6 | 0.7 | 0.8 | 0.75 | M |
| 7 | 0.6 | 0.8 | 0.7 | M |
| 8 | 0.8 | 0.9 | 0.85 | A |
| 9 | 0.9 | 0.9 | 0.9 | A |
| 10 | 0.5 | 0.7 | 0.6 | M |

La selección final de los expertos tuvo lugar atendiendo a los siguientes criterios de interpretación del coeficiente de competencia (k):

Si $0.8 \leq k \leq 1.0$, el coeficiente de competencia es alto.

Si $0.5 \leq k < 0.8$, el coeficiente de competencia es medio.

Si $k < 0.5$, el coeficiente de competencia es bajo.

A pesar de que no resultó ningún especialista con coeficiente de competencia bajo, se seleccionaron ocho teniendo en cuenta el grado científico y los años de experiencia en el desarrollo de software y en seguridad de aplicaciones (Ver figuras 12, 13 y 14).


**Figura 13. Expertos por experiencia en desarrollo de *software*.**

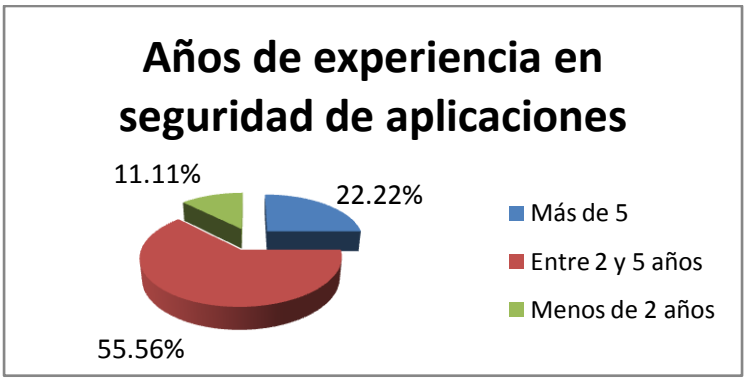


**Figura 14. Expertos por experiencia en temas de seguridad de aplicaciones.**

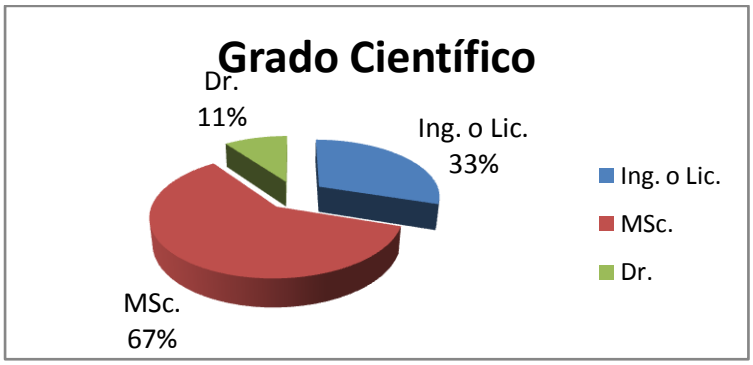


**Figura 15. Expertos por grado científico.**

## 3.3.Confección del instrumento

Para determinar los aspectos a valorar se utilizó la experiencia y conocimientos de los expertos. La documentación de la guía estuvo a disposición de los especialistas con la posibilidad de aclarar dudas al respecto.

En una primera vuelta, se les solicitaron ideas para definir los posibles parámetros de evaluación, a partir de la propuesta que aparece en el Anexo 3. Las respuestas debían estar enmarcadas en una escala numérica representando a las calificaciones: Muy bueno, Bueno, Regular, Pobre y Malo.

Debido a las sugerencias de la inclusión de nuevos parámetros se elaboró una propuesta más completa (ver Anexo 4), que se envió nuevamente a los expertos para su valoración.

## 3.4.Resultados de la evaluación a través del método Delphi

Para verificar la consistencia en el trabajo de expertos, se utiliza el coeficiente de concordancia de Kendall (k) y el estadígrafo Chi cuadrado ($x^2$).

Dados n el número total de criterios a evaluarse y m el número de expertos involucrados en la evaluación, se realiza el siguiente procedimiento para determinar la consistencia del trabajo de los expertos:

1. Calcular para cada criterio la sumatoria del peso dado por cada experto ($C_j$), mediante la expresión: $\sum_{j=1}^{m} C_j$
2. Determinar el valor de puntuación promedio de cada criterio: $P_j$
3. Se calcula el peso medio dado por cada experto $P_i = \frac{\sum_{i=1}^{n} C_i}{n_i}$ y luego $\bar{C} = \frac{\sum_{i=1}^{m} P_i}{m}$
4. Determinar la desviación de la media y elevar el resultado al cuadrado para obtener la dispersión, dada por la expresión: $\sigma^2 = \sum_{j=1}^{n} (\sum_{i=1}^{m} C_i - \bar{C})^2$
5. Conociendo la dispersión se puede calcular el coeficiente de concordancia de Kendall dado por la expresión: $k = \frac{12\ \sigma^2}{m^2(n^3-n)}$
6. Calcular el Chi cuadrado real a partir del valor del coeficiente de Kendall teniendo en cuenta la siguiente expresión: $x^2 = m(n-1)k$

Después de recibir los valores del peso relativo de cada criterio se construye la tabla de los pesos otorgados, para proceder con los cálculos (Ver Tabla 5 en el Anexo 6). De donde se obtuvo que $X^2$ =1.0209.

Para que exista concordancia en el trabajo de los expertos, debe cumplirse que $X^2_{real} < X^2_{tabla}$ [53]. Según la Tabla de Distribución Chi Cuadrado, para α=0.01 y 4 grados de libertad, $X^2_{tabla}$ =13.2767.

Al cumplirse que 1.0209 < 13.2767, se llega a la conclusión de que existe concordancia entre los expertos y no es necesario realizar otra iteración.

Conociendo el peso relativo de cada criterio y la calificación cuantitativa dada por los expertos a cada criterio en una escala de 1 a 5 (Malo, Pobre, Regular, Bueno y Muy bueno), se determina el índice de aceptación (IA) de la propuesta, dado por la expresión:

$$IA = P_j * c/5$$

Donde:

$P_r$: peso relativo de cada criterio

c: calificación cuantitativa dada por los expertos a cada criterio en una escala de 1 a 5

Para realizar este análisis se construye la Tabla de Richman (Ver Tabla 4):

**Tabla 4. Tabla de Richman**

| | 1 | 2 | 3 | 4 | 5 | $P_r$ | $P_r$ x c |
|---|---|---|---|---|---|---|---|
| C1 | | | | | x | 0.07 | 0.35 |
| C2 | | | | x | | 0.06 | 0.25 |
| C3 | | | | | x | 0.08 | 0.38 |
| C4 | | | | | x | 0.08 | 0.40 |
| C5 | | | | x | | 0.06 | 0.25 |
| C6 | | | | | x | 0.07 | 0.36 |
| C7 | | | | | x | 0.08 | 0.39 |
| C8 | | | | | x | 0.07 | 0.37 |
| C9 | | | | | x | 0.07 | 0.35 |
| C10 | | | | | x | 0.07 | 0.37 |
| C11 | | | | x | | 0.06 | 0.24 |
| C12 | | | | | x | 0.07 | 0.35 |
| C13 | | | | | x | 0.08 | 0.40 |
| C14 | | | | | x | 0.08 | 0.40 |
| $P_r * c$ | | | | | | | 4.8158 |
| $P_r * c/5$ | | | | | | | 0.9632 |

El IA es de 0.9632 por lo que se puede afirmar que existe una alta probabilidad de éxito. Teniendo en cuenta que:

IA > 0.7: Alta probabilidad de éxito

0.7 > IA > 0.5: Probabilidad media de éxito

0.5 > IA > 0.3: Probabilidad de éxito baja

0.3 > IA: Fracaso seguro

## 3.5.Aplicación de la propuesta en proyectos reales

Actualmente, la Dirección General de Producción de la UCI, estudia una estrategia efectiva que permita generalizar, en los proyectos de desarrollo de *software,* el uso de la “Guía metodológica para implementar la seguridad durante el desarrollo de aplicaciones informáticas".

## Conclusiones parciales

Al aplicar el método multicriterio para evaluar la factibilidad de aplicar el procedimiento que se propone, se obtuvieron resultados que califican la solución con una alta probabilidad de éxito.

El resultado de este trabajo de investigación será puesto en práctica en proyectos reales, para fortalecer la seguridad de los productos de software desarrollados en la UCI.

# CONCLUSIONES GENERALES

El objetivo de la investigación fue cumplido al obtener una guía metodológica de fácil aplicación, que no requiere de un equipo de especialistas en seguridad para poner en prácticas las actividades que define con el fin de implementar la seguridad en una aplicación.

La propuesta tiene en cuenta todas las etapas del ciclo de desarrollo del software y es suficientemente flexible como para ser aplicable a cualquier entorno de desarrollo.

Para documentar todo el proceso se elaboraron plantillas que contienen descripciones detalladas y ejemplos prácticos.

Para generalizar la propuesta se seleccionó una plataforma que permite el trabajo colaborativo y además de la información relativa a la guía para implementar la seguridad, permite publicar herramientas y complementos para desarrollar las tareas.

# RECOMENDACIONES

Al concluir esta investigación, se recomienda para el desarrollo de estudios futuros:

- Desarrollar herramientas que automaticen las tareas que propone esta guía metodológica.
- Aplicar la propuesta en un pequeño grupo de proyectos reales que se desarrollan en la UCI y paulatinamente incrementar el número de estos a medida que se vaya perfeccionando el proceso.
- Convertir el portal donde se encuentra publicada la Guía metodológica para implementar la seguridad durante el desarrollo de aplicaciones informáticas, en una red temática donde los equipos de desarrollo puedan encontrar toda la documentación que necesitan para fortalecer la seguridad de sus aplicaciones y poner en práctica las tareas de la Guía.

# BIBLIOGRAFÍA


1. ALBET INGENIERÍA Y SISTEMAS. *Plantilla de proyecto técnico*. UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS, ALBET.
2. BRAD HILL. *Cumplimiento en seguridad como disciplina de ingeniería*. *MSDN Magazine*.
3. CARLOS BACHMAIER JOHANNING. *Riesgo (Risk): Gestión. Evaluación, pasos iniciales*. *Dintel. Auditoría y Seguridad*, 2010. Volúmen 5.
4. CMU SEI CERT. *OCTAVE*, 2008. [2011]. Disponible en: http://www.cert.org/octave/
5. DEEPAK MANOHAR. *Threat Modeling*. *Microsoft Security Summits*. New York City, United States, 2006.
6. DEPARTMENT OF HOMELAND SECURITY (DHS). *The Build Security In Software Assurance Initiative (BSI)*, The MITRE Corporation. National Cyber Security Division, 2009. [2011]. Disponible en: https://buildsecurityin.us-cert.gov/bsi/home.html
7. ---. *The Common Attack Pattern Enumeration and Classification (CAPEC)*, The MITRE Corporation. National Cyber Security Division, 2011. [2011]. Disponible en: http://capec.mitre.org/data/index.html
8. DHARMESH M MEHTA. *Effective Software Security Management choosing the right drivers for applying application security*. The Open Web Application Security Project, 16 p.
9. DIRECCIÓN TÉCNICA. *Dictamen sobre el estado de elaboración del Artefacto: Diseño de la Arquitectura de Software v1.2*. *Dictamen de Software*.
10. ESPECIALISTAS DE LA FACULTAD 2. *Plataforma de Gestión de Seguridad Informática. Módulo Pruebas de Intrusión.*: Centro de Calidad para soluciones tecnológicas (CALISOFT) y el Centro de Telemática (TLM), Universidad de las Ciencias Informáticas, 2011. 80. p.
11. ---. *Plataforma para el Análisis Estático de Vulnerabilidades del Código. Distribución y Ejecución de Herramientas.*: Centro de Calidad para soluciones tecnológicas (CALISOFT) y el Centro de Telemática (TLM), Universidad de las Ciencias Informáticas, 2011. 80. p.
12. ESPECIALISTAS DE LA FACULTAD 4. *Procedimiento para Pruebas de Penetración en Aplicaciones Web*. Centro de Calidad para soluciones tecnológicas (CALISOFT) y el Centro de Telemática (TLM), Universidad de las Ciencias Informáticas, 2011. 80. p.
13. ESPECIALISTAS DEL PROYECTO ERP. *Vista de arquitectura de seguridad del proyecto ERP-Cuba*. Proyecto ERP-Cuba, Universidad de las Ciencias Informáticas, 2010. 82. p.
14. FERNANDO APARICIO. *Análisis y gestión del riesgo.*: *IV Jornada Internacional del ISMS Forum SPAIN*, 2008.

15. GARY MCGRAW. *Software Security: Building Security In*. 2006. 448 p. ISBN-10: 0321356705 | ISBN-13: 978-0321356703

16. IEEE COMPUTER SOCIETY PROFESSIONAL PRACTICES COMMITTEE. *Guide to the Software Engineering Body of Knowledge*. 2004. 202 p.

17. INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. *IEEE Standard 1074 for Developing Software Life Cycle Processes*. 1998. p.

18. ---. *ISO/IEC 12207 : 1995*. New York, USA, 1998. p. *IEEE/EIA Standard.* 0-7381-0428-0, SS94581

19. IVAN FLECHAIS AND CECILIA MASCOLO AND M. ANGELA SASSE. *Integrating Security and Usability into the Requirements and Design Process*. *Second International Conference on Global E-Security*. London, UK, 2006.

20. JAMES W. OVER. *Team Software Process for Secure Software Development* 2002. 35 p.

21. JEROME H. SALTZER and MICHAEL D. SCHROEDER. *The Protection of Information in Computer Systems*. IEEE, IEEE

22. JOHN VIEGA and GARY MCGRAW. *Buiding Secure Software: How to Avoid Security Problems the Right Way* 2001. p. *Addison-Wesley Professional Computing.* ISBN-10: 0321774957 | ISBN-13: 978-0321774958

23. JULIA H. ALLEN; SEAN BARNUM*, et al. Software Security Engineering: A Guide for Project Managers*. 2008. 300 p. ISBN-10: 032150917X | ISBN-13: 978-0321509178

24. KAREN MERCEDES GOERTZEL; THEODORE WINOGRAD*, et al. Software Security Assurance: A State-of-the-Art Report (SOAR)*. 396 p.

25. KEN SCHWABER and MIKE BEEDLE. *Agile Software Development with Scrum (Series in Agile Software Development)*. Prentice Hall; 1 edition, 2001. 158 p. ISBN-10: 0130676349 | ISBN-13: 978-0130676344

26. KONSTANTIN BEZNOSOV and P. KRUCHTEN. *Towards Agile Security Assurance*. New security paradigms, Nova Scotia, Canada, 2004. 8 p.

27. LAROUSSE. *Diccionario Enciclopédico Vox 1*. LAROUSSE, 2009.

28. LUIS FRAILE. *Microsoft Solutions Framework Agile*. SYSTEM, M. T.

29. MARK BELK; MATT COLES*, et al. Fundamental Practices for Secure Software Development*. 2nd. 2011. p.

30. MARK G. GRAFF and KENNETH R. VAN WYK. *Secure Coding: Principles & Practices*. O'Reilly, 2003. 224 p. 0-596-00242-4

31. MASS SOLDAL LUND; BJORNAR SOLHAUG*, et al. Model-Driven Risk Analysis: The CORAS Approach*. 2010. 400 p.

32. MEDIAWIKI SOFTWARE. *MediaWiki.org*. Disponible en: http://www.mediawiki.org/wiki/MediaWiki/es

33. MICHAEL HOWARD. *Expert Tips for Finding Security Defects in Your Code*. p.

34. ---. *Inside the Secure Windows Initiative*, 2000. [2011]. Disponible en: http://technet.microsoft.com/en-us/library/cc723542.aspx

35. MICHAEL HOWARD and DAVID LEBLANC. *Writing Secure Code*. 2nd. 2003. 750 p. ISBN-10: 0735617228 ISBN-13: 978-0735617223

36. MICHAEL HOWARD and STEVE LIPNER. *El ciclo de vida de desarrollo de seguridad de Trustworthy Computing*, 2005.

37. ---. *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software*. Microsoft Press, 2006. 304 p. 0-7356-2214-0 978-0-7356-2214-2

38. ---. *The Trustworthy Computing Security Development Lifecycle*. *Annual Computer Security Applications Conference*. Tucson, Arizona (EE.UU.), Microsoft Corporation, 2004.

39. MICROSOFT CORPORATION. *The STRIDE Threat Model*, 2005. [2011]. Disponible en: http://msdn.microsoft.com/library/default.asp?url=/library/enus/csvr2002/htm/cs_se_securecode_zlsj.asp

40. MICROSOFT CORPORATION and CMP MEDIA. *Threat Model Your Security Risks*. *MSDN Magazine. The Microsoft Journal for Developers*, November 2003.

41. MIGUEL ÁNGEL HERNÁNDEZ RUIZ. *CVSS + ISO 27001. Usando Common Vulnerability Scoring System junto a ISO 27001*, 2010.

42. MOHAMMAD REZA AYATOLLAHZADEH SHIRAZI; POOYA JAFERIAN, *et al. RUPSec: An Extension on RUP for Developing Secure Systems*. Transactions on Engineering, Computing and Technology v4, World Enformatika Society 2005. 5 p. 1305-5313

43. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Guide for Developing Security Plans for Federal Information Systems*. NIST, 2006.

44. ORACLE CORPORATION. *Oracle Software Security Assurance*, 2010]. Disponible en: http://www.oracle.com/us/support/assurance/index.html

45. PETER VINCENT HERZOG. *OSSTMM 3.0 LITE. Introduction and sample to the Open - Source Security Testing Methodology Manual*. 2008. 52 p.

46. PITTSBURGH, P. C. S. C. *OCTAVE*, 2008. [Disponible en: http://www.cert.org/octave/

47. PRACTICAL THREAT ANALISYS. *PTA Technologies, Practical Threat Analysis for Securing Computerized Systems.* Disponible en: http://www.ptatechnologies.com/

48. ROBERT C. MARTIN. *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall; 1st edition, 2002. 529 p. ISBN-10: 0135974445 | ISBN-13: 978-0135974445

49. ROBERT C. SEACORD. *Best Practices for Secure Coding (CoBaSSA 2005)*. CERT Coordination Center, 2005. p.

50. ROLANDO ALFREDO HERNÁNDEZ LEÓN and SAYDA COELLO GONZÁLEZ. *El proceso de investigación científica*. Editorial Universitaria del Ministerio de Educación Superior, 2011. 110 p. 978-959-16-1307-3

51. SHANE WARDEN and JAMES SHORE. *The Art of Agile Development*. O'Reilly Media, 2008. 430 p.

52. SHAWN HERNAN; SCOTT LAMBERT*, et al. Uncover Security Design Flaws Using the STRIDE Approach*. *MSDN Magazine*, 2006.

53. SIDNEY SIEGEL and N. JOHN CASTELLAN. *Estadistica No Parametrica: Aplicada a las ciencias de la conducta*. Trillas, 1994. 437 p. ISBN-10: 9682451019 | ISBN-13: 978-9682451010

54. SODIYA ADESINA SIMON; ONASHOGA SADIA ADEBUKOLA*, et al. Towards building secure software systems*. Second International Conference on Global E-Security, London, UK, 2006. p.

55. SOURCEFORGE.NET. *The CORAS Method*, 2010]. Disponible en: http://coras.sourceforge.net/

56. STRATEGIC INITIATIVES BRANCH OF THE NATIONAL CYBER SECURITY DIVISION (NCSD) OF THE DEPARTMENT OF HOMELAND SECURITY. *The Build Security In Software Assurance Initiative (BSI)*, 2009. [Disponible en: https://buildsecurityin.us-cert.gov/bsi/home.html

57. THE MITRE CORPORATION. *2010 CWE/SANS Top 25 Most Dangerous Software Errors*. *Common Weakness Enumeration*. CHRISTEY, S., 2010.

58. THE OPEN WEB APPLICATION SECURITY PROJECT. *CLASP. Comprehensive Lightweight Application Security Process v2.0*. Fundación OWASP, 2006. 505 p.

59. ---. *Comunidad libre y abierta sobre seguridad en aplicaciones*, Fundación OWASP, 2010. [2011]. Disponible en: https://www.owasp.org/index.php/Main_Page

60. ---. *Guía de pruebas OWASP*. Fundación OWASP, 2008. 372 p.

61. ---. *A Guide to Building Secure Web Applications and Web Services*. Fundación OWASP, 2006. 293 p.

62. ---. *OWASP Top 10-2010. Los diez riesgos más importantes en aplicaciones Web*. Fundación OWASP, 2010. 22 p.

63. VICTOR A. RODRIGUEZ. *SPSMM. Security Programming Standard Methodology Manual*, 2002.

64. WEB APPLICATION SECURITY CONSORTIUM. *Web Application Security Consortium: Threat Classification*, WASC, 2004. [2007]. Disponible en: http://www.webappsec.org

# GLOSARIO

Amenaza: Un evento de posible ocurrencia que podría dañar o comprometer un activo o un objetivo estratégico.

Arquitectura de seguridad: Descripción detallada de todos los aspectos relativos a la seguridad de un sistema (o red), incluye un grupo de principios para guiar el diseño de la seguridad. La arquitectura describe cómo todos estos aspectos se integran para satisfacer los requerimientos de seguridad.

Ataque: Una acción que se sirve de una o varias vulnerabilidades para materializar una amenaza.

Auditoría: La auditoría constituye un conjunto de procedimientos y técnicas para evaluar y controlar total o parcialmente un sistema informático con el fin de proteger sus activos y recursos, verificar si sus actividades se desarrollan eficientemente de acuerdo con las normas informáticas y generales existentes en cada empresa y para conseguir la eficacia exigida en el marco de la organización correspondiente.

Autenticación: La autenticación es el proceso de validación de la identidad digital de usuarios o procesos.

Autorización: La autorización sucede después de la autenticación y usa atributos o derechos asociados con la identidad digital para determinar a qué recursos puede acceder.

Caso de abuso: Describe al atacante o usuario malicioso y como éste hace, intencionalmente, mal uso del sistema.

Confidencialidad: Sólo los "usuarios" debidamente autorizados deben tener acceso a los datos y/o los recursos.

Disponibilidad: Los recursos deben poder ser accedidos cuando son necesarios y funcionar a un nivel aceptable.

Incidente: Evento no deseado que podría dañar o reducir el valor de los activos.

Integridad: Sólo los usuarios debidamente autorizados pueden modificar los datos y/o los recursos de modo apropiado y controlado.

No repudio: Se basa en el hecho de que las acciones realizadas por los usuarios deben quedar registradas de un modo tal que estos no puedan negar posteriormente el haberlas realizado.

Riesgo: la probabilidad (cuantificable) de sufrir una pérdida debido a una amenaza materializada. Su valor depende de dos factores: la frecuencia con la que ocurre la amenaza; el impacto del daño que puede causar.

Vulnerabilidad: Una vulnerabilidad es un punto débil que de explotarse con éxito puede llegar a originar la consecución de una amenaza.

## ANEXO 1. Encuesta: Diagnóstico sobre el tratamiento de la seguridad del software.

Objetivo: Determinar los porcientos de aplicación de algunas de las medidas de seguridad del software por cada fase del ciclo de vida.

**1. Requisitos**

1. ¿En la especificación de requerimientos se tienen en cuenta los principios de seguridad del software más significativos?

   Si ◉ No ○ N/A ○

2. ¿Los requerimientos de seguridad han sido objeto de análisis antes de incluirse en las especificaciones?

   | Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
   |---|---|---|---|---|---|
   | ○ | ○ | ○ | ○ | ○ | ○ |

3. ¿Los requerimientos de seguridad son contemplados en los patrones de ataques, uso indebido/casos de abuso y otros medios específicos del modelado y análisis de riesgos?

   | Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
   |---|---|---|---|---|---|
   | ○ | ○ | ○ | ○ | ○ | ○ |

---

**2. Arquitectura y diseño**

4. ¿Los componentes del diseño arquitectónico están sujetos al análisis y medición de la superficie de ataque?

   | Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
   |---|---|---|---|---|---|
   | ○ | ○ | ○ | ○ | ○ | ○ |

5. ¿Los componentes del diseño arquitectónico están sujetos al análisis de riegos arquitectónicos?

   | Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
   |---|---|---|---|---|---|
   | ○ | ○ | ○ | ○ | ○ | ○ |

6. ¿Los controles de alto valor de seguridad son contemplados por los patrones de diseño de seguridad?

| Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | ○ |

**3. Implementación**

7. ¿Los componentes de software son objeto de análisis de código estático y dinámico, frente a vulnerabilidades y debilidades conocidas?

| Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | ○ |

8. ¿Los defectos descubiertos durante la implementación fueron inyectados en la arquitectura y diseño y durante la especificación de requisitos?

| Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | ○ |

---

**4. Prueba**

9. ¿Los defectos detectados durante las pruebas, fueron inyectados durante la especificación de requisitos?

| Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | ○ |

10. ¿Los defectos detectados durante las pruebas, fueron inyectados en la arquitectura y el diseño?

| Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | ○ |

11. ¿Los defectos detectados durante las pruebas, fueron inyectados en la implementación?

| Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | ○ |

12. ¿Los componentes de software demuestran satisfacción de los requisitos de seguridad, representada por una serie de métodos de prueba (funcional, basada en riesgo, penetración, caja negra, caja blanca, etcétera)?

| Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | ○ |

13. ¿Los componentes de software demuestran niveles requeridos de resistencia y capacidad de recuperación, cuando se someten a patrones de ataque, casos de abuso y otros medios específicos del modelado y análisis de riesgo?

| Todos | La mayoría | Algunos | Muy pocos | Ninguno | N/A |
|---|---|---|---|---|---|
| ○ | ○ | ○ | ○ | ○ | ○ |

## ANEXO 2. Encuesta: Nivel de experiencia de los expertos.

Colega:

Teniendo en cuenta su experiencia profesional usted ha sido seleccionado para colaborar con una investigación encaminada a proponer una guía metodológica para implementar la seguridad en aplicaciones informáticas que se desarrollan en la Universidad de las Ciencias Informáticas.

Para valorar sobre el nivel de experiencia que Ud. Posee, le pedimos realice la siguiente autoevaluación.

**Nombre y Apellidos:**
**Cargo que ocupa:**
**Grado científico:**
**Número de veces que ha sido seleccionado como experto:**
**Años de experiencia en el desarrollo de software:**
**Años de experiencia en temas de seguridad de aplicaciones:**

Marque con una cruz (x), en la casilla que le corresponda al grado de conocimientos que usted posee acerca del tema de investigación que desarrollamos, valorándolo en una escala de 0 a 10 (considerando 0 como no tener absolutamente ningún conocimiento y 10 el de pleno conocimiento de la problemática tratada).

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

Autovalore el grado de influencia que cada una de las fuentes que le presentamos a continuación, ha tenido en su conocimiento y criterios sobre el tema de la presente investigación.

| Fuente de argumentación | Alto | Medio | Bajo |
|---|---|---|---|
| Análisis teóricos realizados | | | |
| Experiencia adquirida | | | |
| Conocimiento de trabajos de autores nacionales | | | |
| Conocimiento de trabajos de autores extranjeros | | | |
| Conocimiento del estado del problema en el extranjero | | | |
| Intuición | | | |

## ANEXO 3. Encuesta: Medidores de los resultados de la investigación.

Estimado Colega:

Usted ha sido seleccionado por la calificación científica-técnica, sus años de experiencia y los resultados alcanzados en su labor profesional, como experto para seleccionar de acuerdo a su opinión, los posibles criterios de evaluación para esta investigación, a partir de la siguiente propuesta.

La evaluación del criterio, significa su valoración respecto a ese parámetro como medidor de la investigación que puede ser Muy bueno, Bueno, Regular, Pobre y Malo. Exprese su opinión en un valor numérico de 1 a 5, siendo 5 Muy bueno y 1 Malo.

Si usted considera que el criterio está mal categorizado, haga una nueva propuesta.

| **Categoría de los criterios** | **Propuesta de peso para la categoría** | **Criterio** | **Evaluación del criterio (de 1 a 5)** | **Otra propuesta de categorización** | **Observación** |
|---|---|---|---|---|---|
| **Científica** | | Calidad de la investigación | | | |
| | | Novedad científica | | | |
| **Necesidad** | | Necesidad de la guía metodológica | | | |
| | | Satisfacción de las necesidades | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | productivas | | | |
| **Aplicabilidad** | | Facilidad de comprensión de la guía metodológica | | | |
| | | Adaptabilidad a diferentes entornos de desarrollo de software | | | |
| | | Facilidad de uso | | | |
| **Nuevas categorías y/o criterios** | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# ANEXO 4. Encuesta: Medidores de los resultados de la investigación (2da Iteración).

Estimado Colega:

Usted ha sido seleccionado por la calificación científica-técnica, sus años de experiencia y los resultados alcanzados en su labor profesional, como experto para seleccionar de acuerdo a su opinión, los posibles criterios de evaluación para esta investigación, a partir de la siguiente propuesta.

La evaluación del criterio, significa su valoración respecto a ese parámetro como medidor de la investigación que puede ser Muy bueno, Bueno, Regular, Pobre y Malo. Exprese su opinión en un valor numérico de 1 a 5, siendo 5 Muy bueno y 1 Malo.

Si usted considera que el criterio está mal categorizado, haga una nueva propuesta.

| Categoría de los criterios | Criterio | Evaluación del criterio (de 1 a 5) |
|---|---|---|
| **Científica** | Calidad de la investigación | |
| | Novedad científica | |
| | Coherencia entre el fundamento teórico y la propuesta | |
| **Necesidad** | Necesidad de la guía metodológica | |
| | Satisfacción de las necesidades productivas | |
| **Aplicabilidad** | Facilidad de comprensión de la guía metodológica | |
| | Adaptabilidad a diferentes entornos de desarrollo de software | |
| | Facilidad de uso | |
| | Impacto | |
| | Cumplimiento de los principios ágiles | |
| **Económico** | Contribución a disminuir los costos de desarrollo | |
| | Garantía de seguridad | |
| **Calidad técnica** | Orden y estructura correcta de la guía | |
| | Relación entre actividades y roles en la guía para cada etapa | |

# ANEXO 5. Encuesta: Valoración de los resultados de la investigación.

Estimado Colega:

Usted ha sido seleccionado por la calificación científica-técnica, sus años de experiencia y los resultados alcanzados en su labor profesional, como experto para evaluar los resultados técnicos de la investigación Guía metodológica para implementar la seguridad en aplicaciones informáticas que se desarrollan en la Universidad de las Ciencias Informáticas. Le agradecemos que ofrezca sus ideas y criterios sobre las bondades, deficiencias e insuficiencias que presenta la guía en su concepción teórica y su futura aplicación práctica.

Sus criterios y opiniones se manejarán de forma anónima. Le agradecemos de antemano su valiosa colaboración.

Debe valorar los aspectos que se relacionan a continuación utilizando una escala de 1 a 5, donde 5 indica el máximo.

| Categoría de los criterios | Criterio | Calificación |
|---|---|---|
| **Científica** | Calidad de la investigación | |
| | Novedad científica | |
| | Coherencia entre el fundamento teórico y la propuesta | |
| **Necesidad** | Necesidad de la guía metodológica | |
| | Satisfacción de las necesidades productivas | |
| **Aplicabilidad** | Facilidad de comprensión de la guía metodológica | |
| | Adaptabilidad a diferentes entornos de desarrollo de software | |
| | Facilidad de uso | |
| | Impacto | |
| | Cumplimiento de los principios ágiles | |
| **Económico** | Contribución a disminuir los costos de desarrollo | |
| | Garantía de seguridad | |
| **Calidad técnica** | Orden y estructura correcta de la guía | |
| | Relación entre actividades y roles en la guía para cada etapa | |

Valoración Final:

Elementos a Suprimir:

Elementos a Mejorar:

Elementos a Añadir:

Categoría Final de la Investigación:

| | |
|---|---|
| | Excelente: Alta novedad científica, con aplicabilidad y resultados relevantes |
| | Bueno: Novedad Científica. Resultados Destacados |
| | Aceptable: Suficientemente bueno con reservas |
| | Cuestionable: No tiene relevancia científica |
| | Malo: No aplicable |

# ANEXO 6. Cálculo del coeficiente de Kendall y Chi cuadrado

**Tabla 5. Tabla 4. Pesos otorgados por los expertos**

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 | $P_i$ | $\overline{C}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **E1** | 4 | 4 | 5 | 5 | 4 | 5 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 4.64 | 36.07 |
| **E3** | 5 | 4 | 4 | 5 | 4 | 4 | 5 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 4.43 | |
| **E4** | 4 | 4 | 4 | 5 | 4 | 4 | 5 | 3 | 5 | 4 | 1 | 4 | 5 | 5 | 4.07 | |
| **E5** | 4 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 4 | 4 | 3 | 4 | 5 | 5 | 4.36 | |
| **E6** | 5 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 4 | 5 | 4 | 4 | 5 | 5 | 4.57 | |
| **E7** | 4 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 4 | 5 | 5 | 5 | 4.64 | |
| **E8** | 5 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 4.79 | |
| **E9** | 4 | 3 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 5 | 5 | 4.57 | |
| $\sum_{i=1}^{m} C_i$ | 35 | 31 | 38 | 40 | 32 | 36 | 39 | 37 | 35 | 37 | 30 | 35 | 40 | 40 | | |
| $P_j$ | 4.375 | 3.875 | 4.75 | 5 | 4 | 4.5 | 4.875 | 4.625 | 4.375 | 4.625 | 3.75 | 4.375 | 5 | 5 | | |
| $\sum_{i=1}^{m} C_i - \overline{C}$ | 1.07 | 5.07 | 1.93 | 3.93 | 4.07 | 0.07 | 2.93 | 0.93 | 1.07 | 0.93 | 6.07 | 1.07 | 3.93 | 3.93 | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $(\sum_{i=1}^{m} C_i - \overline{C})^2$ | 1.15 | 25.72 | 3.72 | 15.43 | 16.58 | 0.01 | 8.58 | 0.86 | 1.15 | 0.86 | 36.86 | 1.15 | 15.43 | 15.43 | | |
| $\sum_{j=1}^{n}(\sum_{i=1}^{m} C_i - \overline{C})^2$ | | | 142.9286 | | | | | | | | | | | | | |
| **Kendall** | | | 0.0098 | | | | | | | | | | | | | |
| **Chi cuadrado** | | | 1.0209 | | | | | | | | | | | | | |