CIMAT

Centro de Investigación en Matemáticas, A.C.

Incorporación de prácticas de seguridad de software en TSPi

T E S I S

Que para obtener el grado de

Maestro en Ingeniería de software

P r e s e n t a

Jorge Alberto Barrios García

Director de Tesis:

Dra. Perla I. Velasco Elizondo

Zacatecas, Zac., Julio de 2010

Tabla de contenido.

Índice de ilustraciones.

Índice de Tablas.

1 Introducción.

El presente reporte técnico trata sobre la incorporación de prácticas de seguridad en el proceso de desarrollo de software. Para ello por un lado se emplea TSPi (Introductory Team Software Process) [1]. TSPi es un framework de desarrollo de software muy completo que cuenta con elementos de proceso tanto para construir como para administrar el trabajo en equipo dentro de las organizaciones de desarrollo de software. Por otro lado se considera a CLASP (Comprehensive Lightweight Application Security Process) [7]. CLASP es un conjunto de componentes de proceso que permite de manera formal incluir las mejores prácticas de seguridad en los productos de software. Así, este trabajo propone la forma de aprovechar las cualidades ofrecidas por el proceso TSPi y a su vez fortalecerlo al incorporar las prácticas de seguridad mediante el uso de CLASP.

1.1 Motivación.

Actualmente muchas de las organizaciones dependen en gran medida de los sistemas informáticos para llevar a cabo sus operaciones. Por ello la información que éstas emplean es considerada un activo estratégico digno de ser protegido. Derivado de esto, resulta una tarea importante el administrar y proteger dicha información de amenazas a la seguridad de manera efectiva.

Similarmente, los sistemas informáticos son muy utilizados en los hogares para muy variadas actividades cotidianas de las personas. Llaman la atención las actividades que involucran cuestiones de comunicación, como por ejemplo las compras por internet. Por esta razón los sistemas informáticos empleados para esas tareas deben contar con mecanismos de seguridad que protejan a los usuarios en materia de confidencialidad de la información.

Dado que el software que se usa en las organizaciones como en los hogares requiere proteger la información que estos sistemas manejan, es necesario el contar con un proceso de desarrollo de software que considere prácticas de seguridad. Idealmente, estas prácticas deberían encontrarse incorporadas de manera ordenada y explícita en el proceso de desarrollo. Así mismo, las prácticas deberían estar bien documentadas y proveer la facilidad de tomar métricas que permitan evaluar su efectividad para fines de mejora.

Por otro lado, se ha venido fomentando en las academias la enseñanza del desarrollo de software por medio de procesos haciendo mención de las ventajas que el uso de éstos proveen, tal como el hecho de que sea repetible, medible y por ende mejorable, por mencionar algunas. Paralelamente a esto también se comienza a inculcar en los alumnos una cultura de seguridad de la información enseñándoles técnicas para dicho fin. Desafortunadamente no solamente de manera paralela, sino totalmente aislada del proceso de desarrollo de software. Esto resulta ser problemático debido a que los estudiantes no tienen claridad sobre el cómo y en qué partes del proceso de desarrollo se deben implementar las prácticas de seguridad aprendidas.

Dado lo anterior, podemos decir que se ha vuelto necesario contar con una propuesta que permita incorporar prácticas de seguridad en el proceso de desarrollo de software de una forma más eficiente.

1.2 Descripción del problema.

TSPi en un proceso operacional diseñado para soportar principios bien establecidos de ingeniería de software. Su principal objetivo es ayudar a los equipos de ingeniería a desarrollar productos de calidad bajo restricciones de presupuesto y calendario de manera rápida y confiable.

Para lograr esto TSPi incorpora las mejores prácticas de ingeniería de software tales como administración correcta de equipos en un proyecto, la administración de procesos, administración de riesgos y métricas de software. Sin embargo, y aún contando con las ventajas antes mencionadas, TSPi no cuenta o al menos no de manera explícita, de qué forma y en qué partes del proceso debe ser darse la inclusión de la seguridad en los productos de software que son desarrollados bajo su esquema.

Se debe mencionar que existen investigaciones al respecto de la adopción de prácticas de seguridad en TSPi por parte del SEI (Software Engineering Institute) [13] bajo la premisa de que un software sin defectos es un software seguro. Pero aún con esto no define claramente en que fases y mediante que prácticas se puede construir un software seguro en el contexto de protección de información.

1.3 Objetivo.

Definir una propuesta para la incorporación de las prácticas de seguridad de software ofrecidas por algún proceso existente y relevante en el contexto de TSPi.

Objetivos particulares:

- Establecer qué criterios debería cumplir un proceso de seguridad para ser incorporado a TSPi.
- Caracterizar algunos de los procesos existentes en materia de seguridad en base a los criterios establecidos para seleccionar el más adecuado para ser incorporado a TSPi.
- Establecer de qué forma es posible integrar las prácticas de seguridad del proceso seleccionado.
- Realizar la propuesta de integración de las prácticas de seguridad del proceso elegido a TSPi.

1.4 Alcance.

Para éste trabajo se definirá la propuesta de integración de las prácticas de seguridad a TSPi para fines académicos.

Se contempla realizar las adecuaciones al los scripts de TSPi correspondientes al ciclo uno.

La evaluación realizada al nuevo proceso enriquecido con prácticas de seguridad es de carácter cualitativo.

1.5 Organización del reporte técnico.

El reporte técnico presentado en éste documento se integra por cinco capítulos, el primero corresponde a la introducción en el cual se describe la motivación, la problemática, el alcance y en sí los objetivos para este trabajo.

El siguiente capítulo tiene la finalidad de contextualizar al lector sobre TSPi, describiendo sus principales conceptos y características, así como de su estructura y flujo.

En estos mismos términos se integra un tercer capítulo pero con una intención dirigida hacia CLASP.

En el capítulo cuarto se muestra la propuesta realizada en este trabajo para la incorporación las prácticas de seguridad ofrecidas por CLASP al contexto de TSPi.

Finalmente se presentan las conclusiones sobre la evaluación de la propuesta realizada y así mismo se comentan las intenciones futuras sobre éste trabajo.

2 El proceso de desarrollo de software TSPi.

Actualmente la mayoría del software es desarrollado por equipos por lo que es necesario contar con un proceso que contemple esta forma de trabajo. TSPi es un framework que provee un énfasis balanceado de sobre procesos, producto y equipo para ser utilizado con fines de aprendizaje.

Esto es, TSPi es una versión académica basada sobre TSP (Team Software Process) [14] que es un proceso estructurado para la construcción y dirección de equipos de ingeniería para equipos de más de 20 ingenieros quienes desarrollan sistemas de software intensivos a gran escala. Para efectos de este trabajo se utilizará la versión propuesta por TSPi, la cual es descrita en el presente capítulo.

2.1 ¿Qué es un proceso de desarrollo de software?

Un *proceso de desarrollo de software* es una secuencia de pasos requeridos para desarrollar o mantener software. Cuando éste proceso se encuentra apropiadamente diseñado y representado, servirá como una guía para soportar el trabajo de los ingenieros de software.

En términos generales, un proceso de desarrollo de software establece un framework técnico y de gestión para la aplicación de métodos, herramientas y recursos humanos a las tareas de elaboración del software. Idealmente la definición de un proceso de desarrollo de software debe de establecer criterios de entrada y salida para cada una de sus fases de manera que los productos de trabajo generados estén controlados adecuadamente.

Un proceso definido provee los siguientes beneficios:

- Permite comunicación efectiva sobre el proceso entre los distintos usuarios del mismo.

- Provee una base precisa para la automatización del proceso.
- Facilita la reutilización de un proceso al tomarlo como base para la definición de uno nuevo.
- Fomenta la evolución del proceso al proporcionar un medio efectivo para el aprendizaje y un fundamento sólido para la mejora del mismo.
- Ayuda a la administración del proceso. Una administración efectiva requiere planes claros y una manera precisa y cuantificable para su medición.

2.2 ¿Qué es TSPi?

TSPi es un framework definido para cursos de ingeniería de software en equipos. Provee un énfasis balanceado sobre procesos, producto y equipo. TSPi está sustentado en la amplia experiencia de la industria en materia de planeación y administración de proyectos de software. TSPi muestra el cómo planear y administrar un proyecto en equipo, define roles para los miembros en donde cada uno de ellos tiene claramente definidas sus responsabilidades.

TSPi está basado en Team Software Process (TSP) que, como se menciono antes, es un proceso diseñado para soportar el desarrollo de proyectos de software a gran escala en los que frecuentemente se invierten varios años para completarlos. Así, TSPi es una versión reducida de TSP que resulta más familiar y fácil de usar para fines académicos.

La figura 1 presenta los componentes de proceso sobre los cuales se fundamenta TSPi, esto es:

Un proceso de construcción de equipos con objetivos comunes y con un alto grado de compromiso y cohesión.

Un proceso de trabajo que rige al equipo a mediante un conjunto de prácticas de ingeniería.

Así mismo, es importante mencionar que un pre-requisito para un equipo que utiliza TSPi es la comprensión de los procesos y habilidades de PSP [4].

Fig. 1. Construcción de equipos en TSPi.

Los grupos de ingeniería utilizan TSPi para aplicar los conceptos integrados de equipo para el desarrollo de sistemas intensivos en software a través de:

- El establecimiento de metas.
- La elaboración de un plan de equipo.
- La definición de las funciones del equipo.
- El desarrollo de productos en varios ciclos.
- El establecimiento de medidas estándares para calidad y rendimiento de equipos y estudiantes.
- La utilización de evaluaciones de equipo y de roles.
- El proporcionar de guías sobre problemas de trabajo en equipo.
- El llevar a cabo evaluación de riesgos.

Lo anterior lleva a una organización que emplea TSPI a construir equipos auto-dirigidos que planean y dan seguimiento de su propio trabajo. Así mismo ayuda a la organización a establecer una práctica de ingeniería madura y disciplinada para producir software confiable.

2.3 Estructura y flujo de TSPi.

TSPi emplea múltiples ciclos de desarrollo para construir el producto final. Como se puede ver en la figura 2, el ciclo uno inicia con una etapa de lanzamiento, en la cual el instructor describe lo concerniente a los objetivos de producto. El equipo entonces sigue el proceso TSPi a través los siete pasos del proceso: estrategia, planeación, requerimientos, diseño, implementación, pruebas y postmortem. Posteriormente se inicia el ciclo dos en donde los ingenieros repiten estos mismos pasos, esta vez mejorando el producto base producido en el ciclo uno. Entonces si hay tiempo y el proyecto lo requiere se pueden adicionar ciclos subsecuentes.

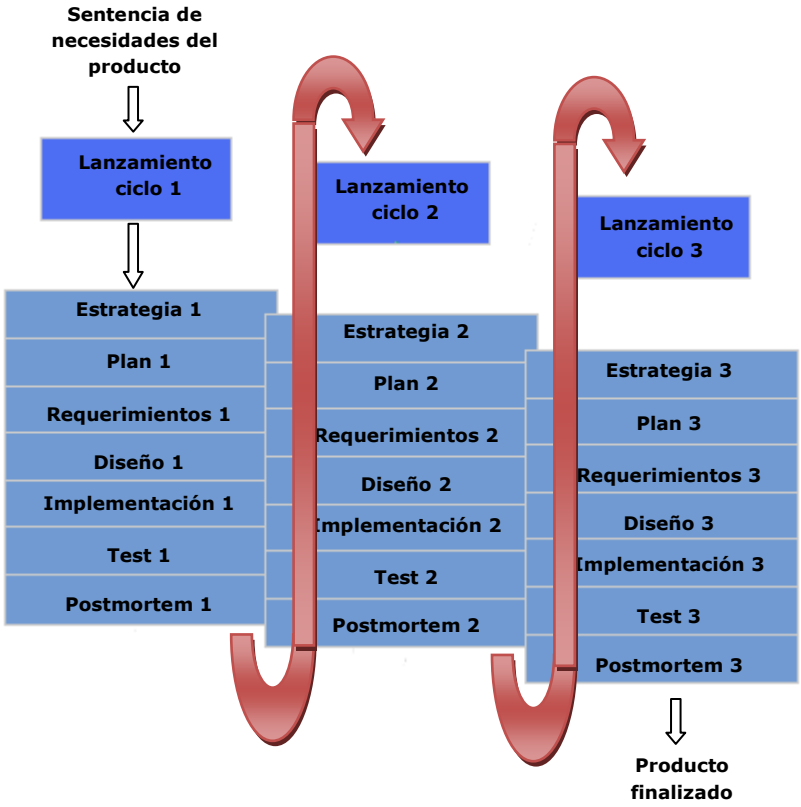

Fig. 2. Estructura y flujo de TSPi.

Al decidir la duración y las actividades a realizar de cada ciclo, se deben considerar las siguientes restricciones:

- En cada ciclo de desarrollo se debe producir una versión que pueda ser probada que sea un subconjunto del producto completo.
- Cada ciclo debería ser lo suficientemente pequeño para hacer claro su desarrollo y probarlo en el tiempo disponible.
- Al reunir los productos generados en cada ciclo se debe obtener el producto final que se planeó.

2.4 Fases del proceso TSPi.

En seguida se describen las fases del flujo del proceso de TSPi.

Lanzamiento.

Hay varias razones para iniciar un proyecto con un lanzamiento. El proceso de formar y construir un equipo no pasa por accidente, y esto toma tiempo. Los equipos necesitan establecer sus interacciones para el trabajo, determinar los roles para los miembros y acordar objetivos. El lanzamiento es el primer paso.

Definir los roles y tomar acuerdos sobre quien manejará cada rol es un paso esencial en la formación del equipo.

Revisar sus objetivos de proyecto, producto y equipo. Los equipos de trabajo deben estar seguros de comprender estos objetivos y entonces usarlos para guiar su trabajo.

Estrategia.

Uno de los aspectos más problemáticos en el desarrollo de software figura en cómo construir sistemas grandes. Estos sistemas son ahora construidos

como una colección de pequeñas partes o módulos para luego ensamblar estas partes. En TSPi se tiene ya una base de decisiones estratégicas en ese sentido para la fase de la estrategia.

El equipo primeramente crea un diseño conceptual para el producto planeado.

El trabajo se debe hacer en un proceso cíclico. Al iniciar con el primer ciclo se diseña, implementa y prueba una versión básica del producto, posteriormente en un segundo ciclo se realiza una nueva versión del producto y, si es necesario y el tiempo lo permite se podría ejecutar un tercer ciclo.

Planeación.

Dependiendo del contexto, los planes pueden ser simples o complejos. La complejidad de un plan está gobernada en gran medida por la complejidad del trabajo que se está planeado realizar. No es una sorpresa entonces que proyectos grandes y complejos frecuentemente requieran planes grandes y complejos. Hay varias razones para hacer planes. Primero cuando se tiene un plan detallado el trabajo se realizará con mayor eficiencia, se sabrá que trabajo hacer y cuando hacerlo. Cuando se planea se tiene una mayor probabilidad de alcanzar las metas

Se debe realizar un plan de calidad. Los procesos de software tienen muchos elementos interrelacionados. Durante la planeación de la calidad se verá como los cambios en tiempo de las fases y la yield por remoción de defectos pueden impactar la calidad del producto. La principal razón para producir el plan de calidad es ilustrar la importancia de un enfoque temprano de la calidad.

Además se debe contar con un balanceo de cargas. Esto no es algo que el equipo debería de realizar y luego olvidarlo. Dependiendo de la precisión de los planes del equipo los ingenieros, incluso tendrían que re-

balancear su carga de trabajo cada semana. Si alguien se queda atrás, el equipo tiene que considerar la opción de brindarle ayuda, incluso si esto significa reubicar a un miembro del equipo de una tarea.

Requerimientos.

Existe frecuentemente confusión sobre la necesidad de un documento de requerimientos. Se asume que la declaración de necesidades describe lo que se quiere y se preguntan por qué un SRS (Software Requirements Specification) sería necesario. La declaración de necesidad, sin embargo, describe lo que quiere el cliente, mientras que el SRS especifica lo que el equipo tiene la intención de construir. El proceso de producción del SRS también ayuda al equipo a estar de acuerdo en lo que planean hacer.

El proceso de requerimientos típicamente inicia con la obtención de requerimientos, en dónde se debe interrogar a los clientes, usuarios y otros stakeholders importantes para descubrir que estos realmente necesitan.

Se debe asegurar la trazabilidad funcional, esto es para cada sección del SRS debe ser posible su identificación con cada ítem de requerimientos. Se inicia a establecer trazabilidad en los requerimientos en la fase de la estrategia y entonces se debe mantener trazabilidad en el SRS y el diseño.

Diseño.

El principal objetivo del diseño es producir, un preciso y completo modelo de solución con fundamentos de alta calidad para la implementación del producto. Un diseño completo define las principales partes del producto, cómo interactúan esas partes y se especifica como estas partes son puestas juntas para producir el resultado.

En la elaboración del diseño para el primer ciclo (de acuerdo a cómo fue dividido el trabajo en la estrategia), los equipos deben anticipar las

mejoras previstas para los ciclos subsecuentes. Si bien aún no se deberían diseñar estas últimas funciones si se deberán proveer tendencias para su posterior inclusión.

Así durante el ciclo inicial el equipo define la arquitectura global del producto pero solo las funciones para el primer ciclo. A partir del diseño de alto nivel puede ser extraordinariamente difícil detectar y corregir defectos durante la implementación y las pruebas. Por lo tanto se debe ser muy cuidadoso con las inspecciones de diseño.

Implementación.

La implementación es una tarea decisiva en el desarrollo del proyecto. Entonces se hace necesaria una buena estrategia para dicha implementación.

La estrategia de implementación debe conformarse generalmente conforme a la estrategia de diseño, esto es, los programas deberían ser implementados consistentemente con la forma en cómo fueron diseñados.

Antes de iniciar la implementación se debe revisar si se ha completado el diseño de alto nivel. Para sistemas grandes éste diseño de alto nivel frecuentemente requiere de varios etapas. En el primer nivel se subdivide el sistema en subsistemas, componentes o módulos. Así mismo se debería tener un diseño detallado.

Pruebas.

La importancia de llevar a cabo pruebas a los programas radica en su propósito, el cual consiste en agregar calidad a los programas producidos en la implementación. Esto es, se verifica que los productos creados sean de alta calidad. Las principales actividades en la fase de pruebas de TSPi son:

Construir el sistema usando las partes desarrolladas a las cuales se les aplicaron pruebas de unidad.

Aplicar pruebas de integración para verificar si el sistema está propiamente construido, es decir que todas las partes están presentes y que éstas funcionan bien juntas.

Llevar a cabo pruebas de sistema para validar que éste hace lo que indican los requerimientos.

Postmortem.

La mejora continua es particularmente importante para los creadores de software debido al tipo de trabajo que se hace. Entonces, el conocer de qué manera y con qué grado calidad se ha desarrollado un software es una habilidad crítica para los ingenieros ya que, deben encontrar oportunidades para mejorar su desempeño y en general sobre el proceso seguido.

Ya se ha descrito que en la fase de estrategia, el desarrollo de un producto en TSPi es realizado mediante ciclos.

Al final de cada ciclo cada equipo escribe un breve resumen sobre su trabajo. Para asegurar que cada equipo produzca reportes similares se debe sugerir un contenido básico para dicho reporte. Como mínimo cada equipo debería comparar su desempeño actual con sus metas para el ciclo y generar una conclusión usando los datos generados de TSPi. Así mismo para datos de planeación del trabajo y calidad del producto.

2.5 Roles del proceso TSPi.

En seguida se describen los roles que participan en el proceso de TSPi. En donde cada rol es responsable de apoyar, administrar y dirigir al equipo de trabajo sobre la realización de las acciones de las cuales es responsable durante la ejecución de las fases de TSPi.

Líder de equipo.

Dirige al equipo y asegura que todos reporten sus datos de las tareas realizadas y completen su trabajo como fue planeado.

Es responsable de motivar al equipo a desarrollar sus tareas, dirigir las reuniones semanales y generar los informes semanales del avance del equipo.

Administrador de desarrollo.

Tiene como objetivo el guiar al equipo a definir, diseñar, desarrollar y probar el producto.

Sus responsabilidades principales son el dirigir la realización de las fases de desarrollo siguiendo los estándares propuestos y generando los productos de cada fase. Integrar el trabajo de todos.

Administrador de planeación.

Apoya y guía al equipo en la planificación y seguimiento del trabajo.

Sus responsabilidades son efectuar la planificación del trabajo de común acuerdo con el equipo de manera balanceada para el siguiente ciclo.

Apoya a producir el calendario para el siguiente ciclo y se asegura que se cumpla. Recaba las mediciones sobre el progreso del equipo.

Administrador de calidad y proceso.

Ayuda al equipo en la definición de necesidades de procesos y en la realización del plan de calidad tanto para el proceso como para el producto.

Es responsable de apoyar al equipo en la definición del proceso, gestionar el plan de calidad, generar estándares para obtener un trabajo

uniforme, moderar las inspecciones y revisiones de los productos generados.

Administrador de soporte.

Apoya al equipo a determinar y conseguir las herramientas necesarias para el trabajo y administra su configuración.

Sus responsabilidades principales son conseguir lo necesario para el desarrollo del proyecto, generar un plan para el control de la configuración y mantener el glosario del sistema.

3 Seguridad en el desarrollo de software.

En el presente capítulo se definen conceptos referentes la seguridad en el desarrollo de software. Posteriormente se realiza una investigación y evaluación de cuatro de los procesos más difundidos en la actualidad en el ámbito de la seguridad en el desarrollo de software. Esto con la finalidad de seleccionar cual sería el más apto para ser incorporado a TSPi.

3.1 ¿Qué es seguridad en el desarrollo de software?

Podemos definir la seguridad en el desarrollo de software como cualquier medida tomada durante la fabricación del software que impida la ejecución de operaciones no autorizadas sobre un sistema. La seguridad evita efectos que puedan conllevar a daños sobre la información, comprometer su confidencialidad, autenticidad o integridad.

Es necesario considerar otros aspectos relacionados cuando se habla de seguridad en el desarrollo de software:

- Cumplimiento de las regulaciones legales aplicables a cada sector o tipo de organización, dependiendo del marco legal de cada país.
- Control en el acceso a los servicios ofrecidos y la información guardada por un sistema informático.
- Control en el acceso y utilización de ficheros protegidos por la ley: contenidos digitales con derechos de autor, ficheros con datos de carácter personal, etcétera.
- Identificación de los autores de la información o de los mensajes.
- Registro del uso de los servicios de un sistema informático, etcétera.

Seguridad de la información.

Desde un punto de vista amplio, en la norma ISO/IEC 17799 se define la seguridad de la información como la preservación de su confidencialidad, su integridad y disponibilidad (medidas conocidas por su acrónimo CIA "Confidentialy, Integrity, Availability"). Esto se representa en la figura 3.

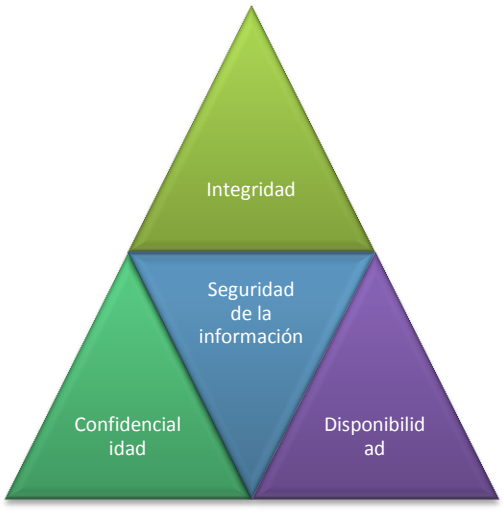

Fig. 3. Seguridad de la información según la norma ISO/IEC 17799.

Dependiendo del tipo de información manejada y de los procesos realizados por una organización, ésta podrá conceder más importancia a garantizar la seguridad de la información.

Debemos tener en cuenta que la seguridad de un sistema informático dependerá de diversos factores, entre los que podríamos destacar los siguientes:

- La sensibilización de los directivos y responsables de la organización que deben ser conscientes de la necesidad de destinar recursos a esta función.
- Los conocimientos, capacidades e implicación de los responsables del sistema informático: dominio de la tecnología utilizada en el sistema informático y conocimiento sobre las posibles amenazas y los tipos de ataques.

- La mentalización, formación y asignación de responsabilidades de todos los usuarios del sistema.
- La correcta instalación, configuración y mantenimiento de los equipos.
- La limitación en la asignación de los permisos y privilegios de los usuarios.
- El soporte de los fabricantes de hardware y software con la publicación de parches y actualizaciones de sus productos que permitan corregir los fallos y problemas relacionados con la seguridad.
- Contemplar no solo la seguridad frente a las amenazas del exterior, sino también las amenazas procedentes del interior de la organización.
- La adaptación de los objetivos de seguridad y de las actividades a realizar a las actividades reales de la organización.

Objetivos de la seguridad informática:

- Minimizar y gestionar los riesgos. Detectar los posibles problemas y amenazas de seguridad.
- Garantizar la adecuada utilización de los recursos y de las aplicaciones del sistema.
- Limitar las pérdidas y conseguir la adecuada recuperación del sistema en caso de un accidente de seguridad.
- Cumplir con el marco legal y los requisitos impuestos por los clientes en sus contratos.

3.2 La seguridad como proceso.

Una organización debe entender a la seguridad en el desarrollo de software como un proceso y no como un producto que se pueda comprar o instalar. Se trata por lo tanto de un ciclo iterativo, en el que se incluyen actividades

como la valoración de riesgos, prevención, detección y respuesta ante incidentes de seguridad. La siguiente figura representa algunas de las actividades fundamentales con las que todo proceso de seguridad en el desarrollo de software debiera contar.

Fig. 4. La seguridad informática como proceso y no como producto.

3.3 Procesos de seguridad en el contexto de desarrollo de software.

CLASP (Comprehensive, Lightweight Application Security Process)

En la documentación consultada acerca de éste proceso [4], CLASP se define como un conjunto de componentes de proceso que permiten a las organizaciones sistemáticamente identificar y mitigar vulnerabilidades que pudieran resultar en fallas en los servicios básicos de seguridad como lo son la confidencialidad, la autentificación y el control de acceso.

CLASP es el resultado de años de trabajo en el que las técnicas aplicadas en muchos ciclos de desarrollo de sistemas fueron identificadas y

descritas a fin de crear un conjunto completo de las mejores prácticas para fomentar la seguridad en los ciclos vida de desarrollos de software tanto para proyectos nuevos como para los ya existentes. Esto hace a CLASP flexible para con los ciclos de vida de desarrollo.

El proceso CLASP está basado en las mejores prácticas y experiencia en el campo de seguridad de empresas de todos los tipos, por lo que resulta ser aplicable tanto para organizaciones pequeñas, medianas y grandes.

En dicha una documentación se describen de manera detallada la definición del proceso en cuanto a roles participantes, las actividades que deberán ser ejecutadas y las vulnerabilidades que deberán ser mitigadas de manera estructurada, repetible y medible.

El proceso de CLASP, como se puede apreciar en la figura 5, está representado a través de cinco perspectivas de alto nivel denominadas vistas que representan componentes del proceso general de CLASP.

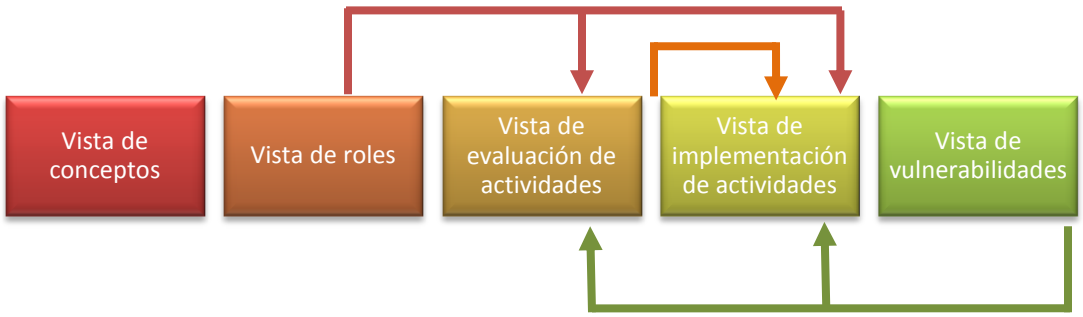

Fig. 5. Proceso de CLASP (simplificado).

En cuanto a la toma de métricas, CLASP propone actividades para la medición de la mejora tanto en el desempeño de los roles, la ejecución de las actividades y para la efectividad en la mitigación de las vulnerabilidades.

Un aspecto que pudiera ser considerado como algo no favorable sobre CLASP es que pudiera resultar complejo su seguimiento si no se tiene cuidado con el orden de la realización de actividades y sobre cómo intervendrán los roles que participan en el proceso, de modo que pudieran presentarse dificultades para que CLASP sea implementado por sí mismo en un proyecto de desarrollo de software. Sin embargo, teniendo un orden bien establecido puede ser implementado de manera exitosa.

MICROSOFT SDL (Software Development Lifecycle)

Microsoft SDL es un proceso que tiene como objetivo garantizar la seguridad en el software al introducir seguridad y privacidad de manera temprana y durante todas las fases del proceso de desarrollo. Es una iniciativa obligatoria en Microsoft desde 2004.

Sobre el proceso SDL se ha publicado solo documentación de contenido muy general [9], en la que se describen brevemente las fases del proceso sin embargo, los detalles y herramientas son tecnología propietaria de Microsoft y por lo tanto no están disponibles.

Los elementos del proceso de seguridad en el proceso de software se muestran enseguida en la figura 6:

Fig. 6. Proceso de seguridad Microsoft SDL

Se puede apreciar que las mejoras en el proceso son incrementales y no requieren cambios radicales en el proceso de desarrollo, aunque no se

especifican métricas para ello, si se hace énfasis en la importancia de realizar mejoras consistentemente en la organización.

Microsoft SLD, aunque en varios de los aspectos en bueno, el hecho de que sea una tecnología propietaria lo hace un proceso no factible para el objetivo perseguido en éste trabajo.

Debido a su estructura y flujo Microsoft SDL resulta ser flexible con respecto de los ciclos de vida de desarrollo de software.

En cuanto a su aplicabilidad en empresas, se comenta en la documentación del proceso [12] que Microsoft SDL podría ser aplicable a todo tipo de empresas. Sin embargo se debe recordar que las herramientas y recursos son solamente disponibles de manera interna a Microsoft.

Software security touchpoints

"Software Security touchpoints" se especifica en un conjunto de touchpoints (actividades destructivas y constructivas. Destructivas sobre los ataques y constructivas sobre el diseño, protección y funcionalidad) que muestran cómo los desarrolladores pueden aplicarlos en los distintos artefactos producidos durante el desarrollo de software. Esto incluye aspectos de requerimientos, arquitectura, diseño, codificación, pruebas, validación, medición y mantenimiento.

Software security touchpoints pretende ayudar a conocer y entender los riesgos comunes para el software, sometiendo a todos los artefactos a un estudio profundo a través de un análisis de riesgos y pruebas.

Algo que se debe comentar es que Software security touchpoints indica las fases que se deben seguir pero no precisa indicaciones sobre el cómo aplicar las prácticas que propone.

Dichas prácticas se muestran a continuación en la figura 7:

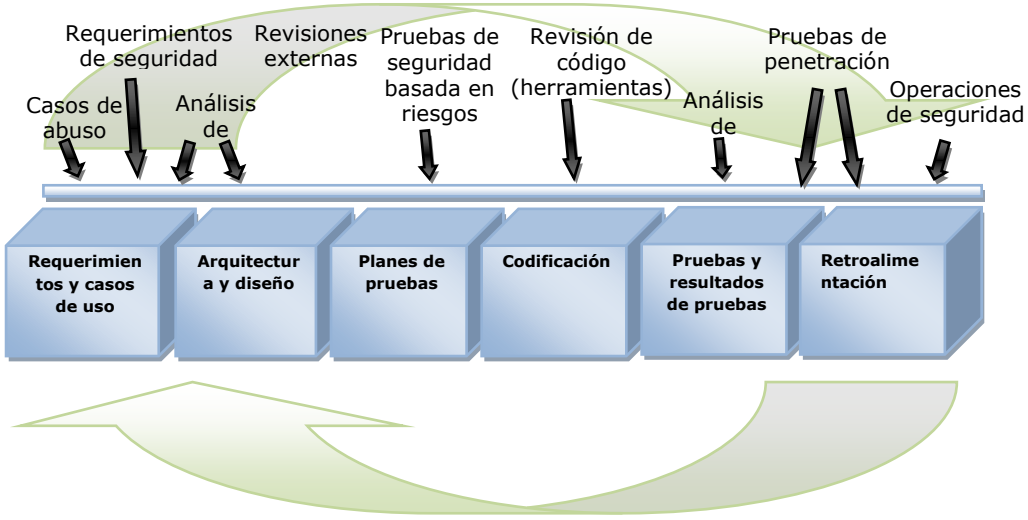

Fig. 7. Proceso Software Security Touchpoints.

En cuanto a la toma de métricas, a pesar de que se menciona su importancia, no se especifica que métricas deben ser tomadas.

Software security touchpoints es un proceso que pretende involucrar explicita y ponderadamente la seguridad en todo el ciclo de vida del software.

En lo que respecta a su aplicabilidad en empresas, no se encontraron datos que indiquen el empleo de éste proceso.

SAMM (Software Assurance Maturity Model).

SAMM es un framework abierto para ayudar a las organizaciones a formular e implementar una estrategia para seguridad en el software, el cual es ajustado a los riesgos específicos a los que se enfrentan. Los recursos proveídos por SAMM ayudan a la construcción de un programa equilibrado de seguridad de software en iteraciones bien definidas.

La base del proceso se fundamenta en que éste se construye sobre las funciones de negocio principales del desarrollo de software con prácticas de seguridad vinculados a cada una de ellas. Los bloques de construcción del proceso se encuentran clasificados en tres niveles de madurez definidos para cada una de las doce prácticas de seguridad.

La siguiente figura muestra la estructura y flujo de actividades del proceso:

Fig. 8. El proceso de SAMM.

SAMM es útil al evaluar las prácticas de seguridad existentes en una organización. También resulta útil para proponer mejoras concretas hacia un programa de garantía de seguridad al definir y medir las actividades relacionadas con la seguridad en una organización.

SAMM de primera instancia resulta ser un proceso muy apropiado y simple, sin embargo tal vez bajo la idea de hacer de SAMM un proceso simple de usar, también se hace que sea un proceso demasiado generalizado para su implantación y un tanto ambiguo para su adaptación a una empresa. Por otro lado considero que para fines de evaluación de la madurez de los procesos de seguridad de una empresa es muy bueno.

En cuanto a los ciclos de desarrollo, no se observa el cómo pueden ser empleados con SAMM debido a que como se comentó anteriormente SAMM resulta bueno para evaluación de la seguridad pero no detalla que forma

pudieran implementarse las prácticas de seguridad durante el desarrollo del software.

En la documentación de SAMM [8] se dice que fue definido pensando en la flexibilidad de modo que pueda ser utilizada por empresas pequeñas, medianas y grandes así como para organizaciones con cualquier estilo de desarrollo.

3.4 Comparación de los procesos de seguridad.

En esta sección se presenta una comparación sobre los procesos de seguridad en el desarrollo de software que fueron analizados para este trabajo, los cuales fueron: CLASP, SAMM, Microsoft SDL y Software security touchpoints.

Para fines de la comparación se establecieron los aspectos que se consideraron relevantes para la elección del proceso que resultara el más apropiado para ser incorporado al proceso TSPi, estos aspectos son los siguientes:

- Flexibilidad con respecto a los ciclos de vida. Esto es identificar si, por su estructura o flujo el proceso de seguridad se puede adaptar a los ciclos de vida de desarrollo.
- Factibilidad para tomar métricas. En este aspecto se determina si el proceso de seguridad se presta para tomar métricas útiles en cuestión de seguridad de software.
- Que tan documentado y definido se encuentra un proceso de seguridad. Esto es identificar si, existe documentación de soporte y se cuenta con scripts y formatos para ser usados en la implementación y seguimiento del mismo.
- Complejidad. Establecer el grado de dificultades que presenta el proceso de seguridad para ser comprendido e implementado.

- Tipos de empresa a las aplica. Este aspecto consiste en determinar si la aplicación del procesos de seguridad puede llevarse a cabo en empresas ya sea pequeñas, medianas o grandes.
- Se establecerá apoyándose en los aspectos anteriores, que tanto es factible que el proceso se incorpore a TSPi.

La siguiente tabla muestra los resultados de la comparación realizada:

Criterio \ Proceso	CLASP	SAMM	MS DSL	TOUCHPOINTS
Flexibilidad/ciclos de vida	Media	Media	Media	Media
Factibilidad/Métricas	Alta	Media	Alta	Media
Definición y documentación del proceso	Alta	Media	Medio	Baja
Complejidad	Media	Media	Alto	Se desconoce
Grado de uso en empresas	Media	Baja	Bajo	Bajo
Tipos de empresas a las que aplica.	Todos los tipos	Todos los tipos	Grande y medianas	Se desconoce
Factibilidad /TSPi	Alto	Medio	Bajo	Bajo

Tabla 1. Resultados de la comparación de los procesos evaluados.

A continuación se describen algunos conceptos sobre el proceso CLASP, y se explican más a detalle los componentes del proceso.

3.5 Conceptos y estructura de CLASP.

La estructura de CLASP y las interacciones entre sus componentes están organizadas en un conjunto de cinco vistas. A continuación se describen cada una de ellas.

Vistas del proceso.

CLASP está representado mediante cinco perspectivas de alto nivel denominadas vistas que son desglosadas como de actividades. Dichas vistas a su vez, contienen componentes de proceso permitiendo rápidamente entender el proceso de CLASP sobre cómo interactúan sus piezas. Dichas vistas son las siguientes, mismas que son mostradas en la figura 10:

- Vista de conceptos;
- Vista basada en roles;
- Vista basada en evaluación de actividades;
- Vista basada en implementación de actividades;
- Vista de vulnerabilidades.

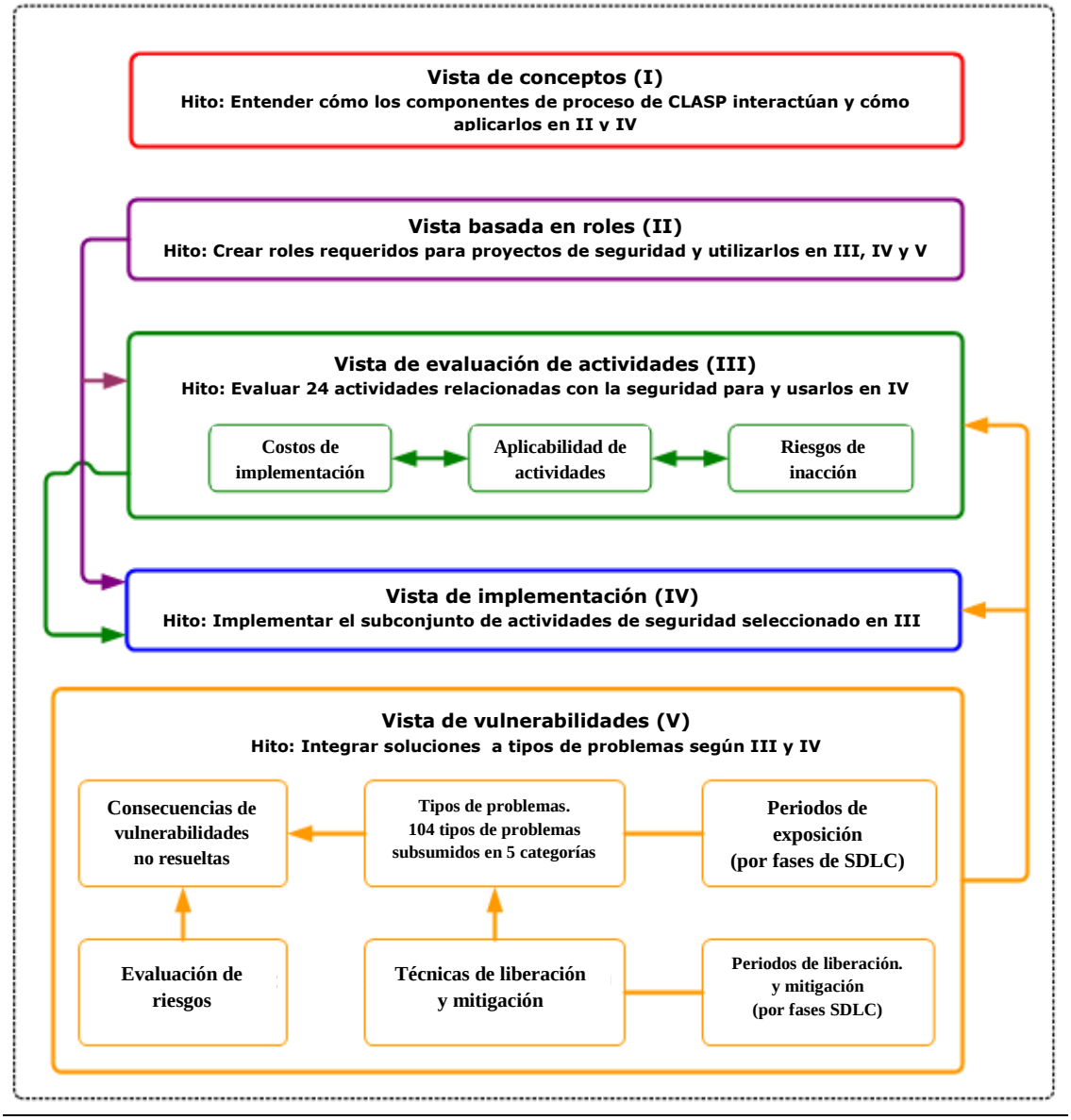

Fig. 9. El proceso CLASP.

Vista de Conceptos.

CLASP se define como un conjunto de componentes de proceso basado en roles y dirigido a actividades. En cuyo núcleo contiene de manera formalizada las mejores prácticas para fomentar la seguridad en los ciclos vida de desarrollo de software. Esto de una manera estructurada, repetible y medible.

La vista de conceptos se desglosa como sigue:

- Descripción del proceso CLASP.
  Provee una descripción de la estructura de CLASP y las dependencias entre sus componentes.

- Mejores prácticas.
  Provee una alternativa razonable para el uso de las mejores prácticas de seguridad en aplicaciones tan pronto como sea posible en todo el ciclo de vida de desarrollo de software.

- Descripción taxonómica.
  En una clasificación de alto nivel del proceso dividida en familias de vulnerabilidades para una mejor evaluación y resolución en materia de seguridad.

Vista Basada en roles.

Esta vista provee una visión de alto nivel a los administradores de proyectos y sus equipos de trabajo sobre cuales actividades de seguridad deben realizar y cuáles son las responsabilidades de cada rol. Estas responsabilidades son mostradas a continuación:

- Administrador de proyectos.
  La primera de ellas es *promover la conciencia*. Usualmente todos los miembros de los equipos deberán tener una introducción a la estrategia de seguridad del software.

  Además debe promover la *concientización fuera del equipo*. El resto de la organización necesita entender acerca de la aplicación de la seguridad en el negocio, tal como ventajas de calendario y riesgos de seguridad. Otra responsabilidad primaria es el *monitoreo de la salud de la organización*.

- Especificador de requerimientos.
  Es el principal responsable de detallar los requerimientos de negocios que son relevantes en cuanto a seguridad. Particularmente en cosas que deberían ser consideradas para la arquitectura.

  Después de que el equipo tiene elaborada una arquitectura candidata debe determinar que requerimientos de protección para los recursos de la arquitectura.

- Arquitecto.
  El arquitecto debería apoyar a la compresión de las implicaciones de seguridad de tecnologías para no introducir errores de seguridad.

  Enumerar todos los recursos en uso por un sistema al nivel más profundo posible.

  Más allá de apoyar en la construcción de los requisitos de seguridad, más bien identificar los roles en el sistema que van a utilizar cada uno de los recursos.

  Identificar las operaciones básicas en cada recurso y estar preparado para ayudar a la gente a entender cómo cada uno de los recursos interactúan con los otros en el sistema.

- Diseñador.
  En primer lugar, debe averiguar qué tecnologías cumplen los requisitos de seguridad e investigar si estos son suficientes para determinar cómo utilizar esas tecnologías correctamente.

  En segundo lugar, si una falla de seguridad es encontrada en la aplicación, por lo general es el diseñador el que debe evaluar las consecuencias y determinar la mejor manera de afrontar el problema.

  Por último, debe contribuir a la tarea de medir la calidad de aplicación y de los esfuerzos de seguridad.

- Implementador.
  Tradicionalmente, el desarrollo de aplicaciones se manejan de manera adhoc, y es el implementador quien tiene la mayor experiencia en seguridad.

- Analista de pruebas.
  Genera y organiza las pruebas necesarias para examinar el cumplimiento de los requerimientos, así como la aplicación de suites de regresión.

- Auditor de seguridad.
  Examinar el estado actual de un proyecto y tratar de garantizar la seguridad del proyecto.

  Al examinar los requerimientos debe determinar si los requerimientos son adecuados y están completos.

  Al evaluar el diseño debe determinar si existe alguna implicación que pudiera producir vulnerabilidades.

  Cuando se examina la implementación tratará de evidenciar defectos de seguridad y estos deben ser mapeados a las desviaciones en las especificaciones.

Vista de evaluación de actividades.

El propósito de esta actividad es aminorar el peso que recae sobre un administrador de proyectos y el equipo de ingeniería de procesos al proporcionar una guía que ayude a evaluar las actividades más apropiadas sobre las 24 actividades propuestas por CLASP:

- Información sobre la aplicabilidad.
  Proporciona información acerca de los tipos de proyectos para los que se recomienda cada actividad de seguridad.

- Una discusión de los riesgos.
  Impacto asociado al omitir actividades de seguridad.

- Indicaciones de los costos de implementación.
  En términos de frecuencia y horas hombre por iteración.

Vista de implementación de actividades.

En el núcleo de CLASP como ya se dijo antes, se tienen contempladas 24 actividades relacionadas con la seguridad que pueden ser integradas en un proceso de desarrollo de software. La actividad de esta fase consiste en llevar a cabo la:

- Implementación.
  Trasladar dentro del software ejecutable el subconjunto de las 24 actividades las cuales fueron evaluadas y aceptadas.

Vista de vulnerabilidades.

El proceso de CLASP cataloga los temas que dan lugar a problemas de seguridad mediante una clasificación de alto nivel o taxonomía, incluyendo la siguiente información sobre cada vulnerabilidad:

- Tipo de problema.
  La noción de tipo de problema coincide con la noción de "causa fundamental", teniendo en cuenta qué cada vulnerabilidad suele estar compuesta por múltiples problemas.

- Categoría.
  Los tipos de problemas están documentados en un conjunto de categorías que los agrupan de manera jerárquica.

- Fase de exposición.
  Se refiere a la etapa en el ciclo de desarrollo de software en la cual la falla puede ser introducida en el sistema.

- Consecuencias.
  Información referente a la consecuencia de caer en determinada vulnerabilidad.

- Plataformas.
  Indicaciones acerca de qué plataformas serían afectadas en términos de lenguajes de programación.

- Recursos.
  Describe con qué recursos debería contar el atacante para recurrir a un ataque a la seguridad.

- Evaluación de riesgos.
  Información acerca de la severidad y la probabilidad de que la vulnerabilidad sea atacada.

- Fase de mitigación y liberación.
  Se refiere a la etapa del ciclo de desarrollo de software en la cual la falla puede ser mitigada y liberada.

4 Incorporación de prácticas de seguridad en TSPi.

El presente capítulo trata sobre la estrategia seguida para llegar a elaborar la propuesta de este trabajo, la cual consiste en la incorporación de prácticas de seguridad a los procesos de TSPi.

A continuación se describen cuales fueron los pasos que se siguieron para lograr el objetivo.

4.1 Descripción de la propuesta.

En el capítulo dos se indica que en este trabajo TSPi es el proceso de desarrollo de software que será tomado como base para la generación del proceso propuesto debido a las ventajas que presenta. Entre dichas ventajas se pueden mencionar su alto grado de documentación, flexibilidad para ser modificado y factibilidad para ser utilizado en un ambiente académico.

Posteriormente en el capítulo tercero, se muestra el resumen de cuatro de los procesos de seguridad en el desarrollo de software más populares (SAMM, CLASP, Software security touchpoints y Microsoft SDL). Cada uno de dichos procesos fue evaluado con respecto de los otros tomando como base los siguientes criterios:

- Flexibilidad con respecto a los ciclos de vida.
- Factibilidad para tomar métricas.
- Qué tan documentado y definido se encuentra un proceso de seguridad.
- Complejidad.
- Tipos de empresa a las que aplica.

Una vez evaluadas se determinó la factibilidad de que el proceso pudiese ser incorporado al proceso TSPi.

Como se puede apreciar en la siguiente tabla, CLASP resultó el proceso que presentó un mayor grado de adaptabilidad a TSPi debido a que, a pesar de resultar medianamente complejo para ser implementado y comprendido, resultó ser el que mejor documentado se encuentra. A demás se presta adecuadamente para la toma de métricas de seguridad y en general es flexible para adaptarse a los ciclos de vida de desarrollo de software.

Criterio \ Proceso	CLASP
Flexibilidad/ciclos de vida	Media
Factibilidad/Métricas	Alta
Definición y documentación del proceso	Alta
Complejidad	Media
Grado de uso en empresas	Media
Tipos de empresas a las que aplica.	Todos los tipos
Factibilidad /TSPi	Alto

Tabla 2. Resultados evaluación CLASP.

4.2 Integración de CLASP a TSPi

Primeramente se analizó en qué fases del proceso de TSPi deberían ser integradas las diferentes vistas del proceso de CLASP. Esto tomando en cuenta las metas de cada fase de TSPi con respecto a las metas de cada una de las vistas de CLASP.

Esto es mostrado en la siguiente tabla:

Vistas CLASP \ Fases TSPi	Lanzamiento	Estrat	Plan	Requer	Diseño	Imple	Pruebas	Post Mortem
Vista de conceptos	X							
Vista basada en roles	X							X
Vista de evaluación de actividades		X	X	X				X
Vista de implementación				X	X	X	X	X
Vista de vulnerabilidades					X	X	X	X

Tabla 3. Integración de las vistas de CLASP a las fases de TSPi.

En la tabla 3 se han relacionado cada una de las vistas del proceso CLASP para indicar en qué partes del proceso de TSPi deben ser integradas.

Lo anterior significa que además de las actividades ya pre-existentes en los scripts del proceso TSPi, se deben integrar de manera organizada las correspondientes del proceso CLASP. (Ver anexo 1).

Estas adecuaciones se detallan continuación:

Se puede observar que la vista de conceptos de CLASP se relaciona únicamente con la fase de lanzamiento de TSPi, esto es debido a que al iniciar con dicha fase se debe de tener como pre-requisito el conocer y entender al proceso CLASP. Sin embargo, dichos conceptos se deben tener presentes durante todas las fases del proceso de TSPi.

Continuando con la fase de lanzamiento, esta debe relacionarse muy estrechamente con la vista de roles de CLASP debido a que es durante esta fase donde se establecen los roles para los miembros del equipo de trabajo y por ende se deben tener a la mano los objetivos y actividades de cada rol.

Ahora bien, durante la fase correspondiente al desarrollo de la estrategia, se deberán seleccionar de las 24 actividades del proceso contempladas por CLASP, las que de acuerdo al tipo y naturaleza del software a desarrollar apliquen de mejor forma. Esto es, se debe contemplar en esta fase la selección de un subconjunto de las tareas que deberán ser integradas en la estrategia de desarrollo de modo que los aspectos de seguridad sean cubiertos de la mejor manera.

Durante la fase de planeación se deberán contemplar el total de tareas de la estrategia para la calendarización y distribución del trabajo en los ciclos que se hayan establecido, incluyendo las de seguridad por supuesto.

Posteriormente en la fase de requerimientos, además de recabar, analizar y validar requerimientos en cuanto a la funcionalidad, se deberán de documentar los requerimientos funcionales y de negocios en cuestión de seguridad. Eso es, documentar supuestos y requerimientos acerca del entorno operativo para que el impacto en la seguridad pueda ser evaluado.

La fase de diseño consiste de, en primer lugar, averiguar qué tecnologías cumplen los requisitos de seguridad requeridos e investigarlas suficientemente para determinar cómo utilizarlas correctamente. En segundo lugar, si una posible amenaza de seguridad es encontrada para la aplicación, se deberán evaluar las consecuencias y determinar la mejor manera de afrontarlas.

En la fase de implementación en gran medida depende de lo que se haya determinado en la fase de diseño y consiste precisamente en implementar todas las actividades de seguridad que se decidió en fases anteriores.

Finalmente en la fase de pruebas se revisa que todas las amenazas y en general todos los requerimientos documentados en cuanto a seguridad están cubiertos. Para ello se deberán diseñar caso de prueba que cubran dichos requerimientos. En CLASP se cuenta con la vista de vulnerabilidades

que contiene una taxonomía de posibles amenazas que puede ser consultada durante el diseño de las pruebas.

4.3 Equivalencias de roles CLASP y roles TSPi.

El proceso CLASP como se mencionó antes, contiene una vista de roles que es una de las cinco vistas de dicho proceso de seguridad. En base a esto y partiendo de las metas y actividades de los roles de CLASP con respecto de los roles de TSPi, se llevó a cabo la tarea de identificar la mejor manera en que las actividades de ambos procesos pudieran ser conjuntadas en los roles de TSPi. Esto es representado en la tabla siguiente:

ROLES TSPi		ROLES CLASP
ADMINISTRADOR DEL PROYECTO		ADMINISTRADOR DEL PROYECTO
LIDER DE EQUIPO		NO HAY CORRESPONDENCIA
ADMINISTRADOR DE DESARROLLO	**ADMINISTRADOR DE REQUERIMIENTOS**	ESPECIFICADOR DE REQUERIMIENTOS
	ADMINISTRADOR DEL DISEÑO	ARQUITECTO
		DISEÑADOR
	ADMINISTRADOR DE DESARROLLO	IMPLEMENTADOR
	PRUEBAS	ANALISTA DE PRUEBAS
ADMINISTRADOR DE PLANEACIÓN		NO HAY CORRESPONDENCIA
ADMINISTRADOR DE CALIDAD Y PROCESOS	**ADMINSITRADOR DE CALIDAD**	AUDITOR DE SEGURIDAD
	ADMINISTRADOR DE PROCESOS	NO HAY CORRESPONDENCIA
ADMINISTRADOR DE SOPORTE		NO HAY CORRESPONDENCIA

Tabla 4. Correspondencia de roles de CLASP y roles TSPi.

En primera instancia tenemos al administrador de proyectos, que aunque no forma parte del equipo de trabajo directamente según los roles del proceso TSPi, éste ha sido considerado en la tabla anterior debido al

grado de importancia que adquiere para el cumplimiento de los objetivos de seguridad al contribuir con sus gestiones y apoyo en general por parte de la organización al equipo de trabajo.

En seguida tenemos al líder de equipo, que aunque no se localizó ninguna correspondencia directa con los roles de CLASP, resulta vital para el monitoreo de las actividades de seguridad y general para el cumplimiento de las metas de equipo de trabajo.

El cuanto al administrador de desarrollo, éste es un rol que en muchas ocasiones dependiendo de la naturaleza del proyecto, durante la fase de lanzamiento se decide descomponerlo en roles más específicos, tal es el caso de los roles de diseño, requerimientos y pruebas. Si este es el caso, la correspondencia de puestos quedaría como está representada en la tabla anterior.

En el caso del administrador de planeación, sus funciones no cambian. Sin embargo debe tener precaución de asignar los recursos suficientes a las actividades concernientes a la seguridad y general con todas las actividades de proyecto.

Acerca del administrador de calidad y procesos, éste rol cumple una tarea preponderante para los proyectos de desarrollo, debido a que es quien se encarga de asegurar la calidad de los productos en todos los aspectos y por supuesto también en cuestión de seguridad. Además, debe fomentar y participar en la toma las métricas correspondientes que permitirán monitorear el cumplimiento de los objetivos fijados para el proyecto, el producto y el equipo.

Finalmente tenemos al administrador de soporte para el cual no existe una correspondencia en los roles de CLASP. No obstante debe quedar entendido que el responsable de este rol, deberá estar capacitado para la instalación, configuración y la administración del equipo que será usado en el desarrollo del software, tanto en cuestiones de seguridad como de cualquier otro aspecto.

4.4 Correspondencia de las actividades de CLASP en las fases TSPi.

A continuación se presentan las 24 actividades del proceso CLASP y su correspondencia referente a la fase de TSPi en la que deben ser llevadas a cabo, lo cual es mostrado en la siguiente tabla. Así mismo se muestran los roles de CLASP que son responsables por coordinar que éstas tareas sean ejecutadas.

Actividad	Rol responsable	Descripción	Lanzam	Estrat	Plan	Requerim	Diseño	Implem	Pruebas
Instituir programas de sensibilización a la seguridad.	Administrador del proyecto	• Asegurar que los miembros del proyecto consideren a la seguridad como un objetivo importante para proyecto mediante la formación y responsabilización. • Asegurar que los miembros del proyecto tengan capacitación y experiencia suficiente sobre la seguridad para tratarla con eficacia.							
Monitoreo de métricas de seguridad	Administrador del proyecto	• Medir el probable estado actual de seguridad del esfuerzo desempeñado. • Exigir responsabilidad por seguridad inadecuada.				X	X	X	X
Especificación del entorno operacional	Especificador de Requerimientos	Documentar supuestos y requerimientos acerca del entorno operativo para que el impacto en la seguridad pueda ser evaluado.				X			
Identificar políticas de seguridad global	Especificador de Requerimientos	• Proporcionar requerimientos del negocio de base por default para la seguridad de productos. • Proporcionar un método de comparación del estado de seguridad de diferentes productos a través de la organización.				X			
Identificar recursos y límites de funciones	Arquitecto	Proporcionar una base estructurada para entender los requerimientos de seguridad de un sistema.				X			
Identificar roles de usuarios y sus capacidades de recursos	Arquitecto	Definir los roles y las capacidades/recursos del sistema a los que el rol puede tener acceso.					X		

Actividad	Rol responsable	Descripción	Lanzam	Estrat	Plan	Requerim	Diseño	Implem	Pruebas
Documentar requerimientos de seguridad relevantes	Especificador de Requerimientos	Documentar los requerimientos funcionales y de negocios en cuestión de seguridad.				X			
Detallar casos de malversaciones.	Arquitecto	Comunicar la razón de ser de decisiones de seguridad relevantes a los stakeholder.					X		
Identificar la superficie de ataque	Diseñador	Especificar todos los puntos de entrada a los programas que facilite el análisis.					X		
Aplicar principios de seguridad al diseño.	Diseñador	Aplicar principios de seguridad al diseño					X		
Investigar y evaluar la situación de soluciones de seguridad.	Diseñador	• Fortalecer el diseño de aplicaciones mediante la aplicación de los principios de seguridad en el diseño. • Determinar las estrategias de implementación para servicios de seguridad. • Diseño de protocolos seguros y API.					X		
Documentar los diseños de clases con propiedades de seguridad	Diseñador	Elaborar políticas de seguridad para campos de datos individualmente.					X		
Especificar la configuración de seguridad en la base de datos.	Diseñador de Base de datos	• Definir una configuración de seguridad por defecto de los recursos de base de datos que se implementan como parte de la implementación. • Identificar una configuración recomendada para los recursos de base de datos para bases de datos que se implementan por terceros.					X		

Actividad	Rol responsable	Descripción	Lanzam	Estrat	Plan	Requerim	Diseño	Implem	Pruebas
Desarrollar análisis de seguridad de los requerimientos y diseño del sistema (modelado de amenazas)	Auditor de seguridad	• Evaluar la probabilidad de riesgos de sistema de manera oportuna y rentable mediante el análisis de los requerimientos y diseño. • Identificar las amenazas de alto nivel del sistema que no están documentados en los requerimientos o documentación complementaria. • Identificar requerimientos inadecuados o incorrectos de seguridad. • Evaluar el impacto a la seguridad de los requerimientos no confiables.	X			X	X		
Integrar el análisis de de seguridad en el manejo del proceso de origen.	Integrador	Automatizar el análisis de seguridad a nivel de implementación y recolección de métricas					X	X	
Implementar contratos de las interfaces	Implementador	• Proporcionar semántica de validación de entradas a nivel de unidad. • Identificar de manera confiable errores de manera estructurada a la mayor brevedad.					X		
Implementar y elaborar políticas de recursos y tecnologías de seguridad.	Implementador	Implementar seguridad funcionalmente con las especificaciones.						X	
Direccionar lo issues de seguridad reportados.	Diseñador	Asegura que los riesgos de seguridad identificados en la implementación sean considerados apropiadamente.						X	
Desarrollar revisiones de seguridad desde su origen	Auditor de seguridad	Encontrar vulnerabilidades de seguridad introducidas en la implementación						X	

Actividad	Rol responsable	Descripción	Lanzam	Estrat	Plan	Requerim	Diseño	Implem	Pruebas
Identificar, implementar y desarrollar pruebas de seguridad.	Analista de pruebas	• Encontrar problemas de seguridad no detectados por la revisión de la implementación. • Encuentra los riesgos de seguridad introducidos por el entorno operacional. • Actuar como un mecanismo de defensa en profundidad, capturando fallos en el diseño, especificación o implementación							X
Verificar atributos de seguridad de los recursos	Analista de pruebas	Confirmar que el software se ajusta a las políticas de seguridad previamente definidas							X
Realizar la firma del código	Integrador	Proporcionar a los stakeholders un medio de validar el origen y la integridad del software.						X	
Construir una guía operacional de seguridad	Integrador	• Proveer a los stakeholders con la documentación sobre medidas de seguridad operacional que pueda mejorar la seguridad el producto. • Proveer documentación para el uso de la seguridad funcional del producto.						X	
Manejar un proceso de divulgación de issues de seguridad	Administrador de proyecto	• Comunicarse efectivamente con los investigadores de seguridad externa cuando las cuestiones de seguridad se identifican en el software liberado. • Comunicarse efectivamente con los clientes cuando los problemas de seguridad se identifican en el SW.					X		

Tabla 5. Correspondencia de actividades CLASP a las fases del proceso TSPi.

5 Evaluación de la propuesta.

Esta sección tiene la finalidad de presentar la evaluación que se llevó a cabo sobre la propuesta de éste trabajo. En ese sentido se han planteado una serie de supuestos acerca de la ya mencionada propuesta (mencionados en la sección siguiente).

El objetivo de ésta evaluación es comprobar si cada uno de los supuestos sobre la propuesta en opinión de quienes serían los usuarios del proceso, son verdaderos.

5.1 Supuestos acerca de la propuesta.

- Debido a que tanto TSPi como CLASP cuentan con suficiente documentación y, basándose en el hecho de que si el proceso TSPi presenta claridad para ser seguido y ejecutado, la propuesta creada en éste trabajo, también debería tener claridad suficiente para ser ejecutada sin demasiados problemas.
- Sobre TSPi se argumenta facilita la toma de métricas que ayudan a madurar el proceso de desarrollo de software. Es entonces fácil suponer que si la propuesta se basa en dicho proceso, debería seguir siendo también factible en este aspecto incluyendo las métricas en materia de seguridad.
- Se ha dicho anteriormente que TSPi es una versión orientada a facilitar el aprendizaje y su adopción en la academia, se supondría que la propuesta presentada en este documento también lo es.
- Tanto TSPi como CLASP se caracterizan por ser lo suficientemente flexibles como para poder adaptarse a cualquier ciclo de vida de desarrollo. Se supone entonces que la propuesta presentada también lo es.

- Una buena planeación de proyecto está basada en gran medida en la estimación del tamaño de los artefactos a desarrollar y por ende el tiempo que se llevará producirlos. Esta estimación permite realizar la correcta calendarización y asignación del trabajo.

  Se puede suponer que al incluir aspectos de seguridad durante la estrategia del proyecto, ayudará a disminuir el error de estimación de los artefactos que serán producidos. Incluso considerar la creación de artefactos nuevos totalmente orientados a la seguridad del software.

- Una última suposición sobre la propuesta es que los roles establecidos en TSPi son suficientes para soportar la carga de trabajo que representa la inclusión de prácticas de seguridad.

5.2 Diseño del instrumento de recolección de información.

Para llevar a cabo la evaluación del proceso propuesto, se empleó como instrumento de recolección de información una encuesta, de la que cabe mencionar que la información que se planeó recolectar es de carácter cualitativo por el momento.

Cada pregunta de la encuesta servirá para medir a cada uno de los supuestos sobre la propuesta de ésta investigación. Así mismo la encuesta solicita a los encuestados escribir las razones que motivaron sus respuestas a manera de sondeo.

5.3 Descripción de marco muestral.

Para la aplicación del instrumento de recolección de información se tomó en cuenta una muestra de 40 personas. Todas las personas encuestadas tienen relación con la carrera de Ingeniería en tecnologías de la

información y comunicación de la Universidad Tecnológica del Estado de Zacatecas: personal de desarrollo de software, docentes y alumnos de octavo cuatrimestre de la materia de calidad en el desarrollo de TI. Considerando así a los usuarios potenciales de la propuesta.

5.4 Aplicación del instrumento de recolección de información.

La estrategia seguida para la aplicación de la encuesta fue la siguiente:

Primeramente se planteó que previo a la aplicación del instrumento, los posibles encuestados deberían conocer los procesos de TSPi y CLASP.

A los docentes y desarrolladores se les proporcionó material de lectura con una semana de anticipación.

En cuanto a los alumnos, TSPi les fue explicado en clase y dejado la actividad de leer la documentación acerca de CLASP.

Posteriormente, se les dio a conocer la propuesta mediante una exposición detallada. Después de estas actividades la encuesta fue aplicada.

5.5 Análisis de datos e interpretación de resultados.

Enseguida se presentan para cada pregunta de la encuesta aplicada, el gráfico representativo de los resultados y la interpretación respectiva de la misma.

Pregunta 1.

¿Piensas qué la propuesta presentada es lo suficientemente clara para ser ejecutada sin demasiados problemas?"

- Gráfico de porcentajes.

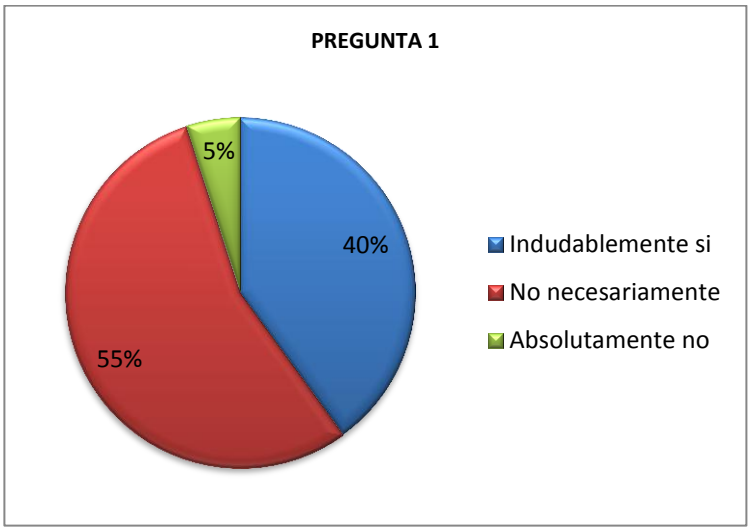

Fig. 10. Gráfica de resultados de la pregunta 1 de la encuesta.

- Interpretación.

Como se puede observar el 55% de los encuestados creen que la incorporación de CLASP a TSPi pudiera no necesariamente ser lo suficientemente clara para poder seguirlo y ejecutarlo. Según los comentarios realizados por éstos, la gran mayoría cree que al unir los procesos, podrían existir discrepancias que podrían hacer confuso el seguimiento de actividades, pero que sin embargo puede irse ajustando con el paso del tiempo. Por otro lado el 40% está completamente convencido de que si sería posible la mencionada incorporación.

Pregunta 2.

¿Piensas qué la propuesta es factible para tomar métricas que ayuden a madurar el proceso de desarrollo de software en materia de seguridad?"

- Gráfico de porcentajes.

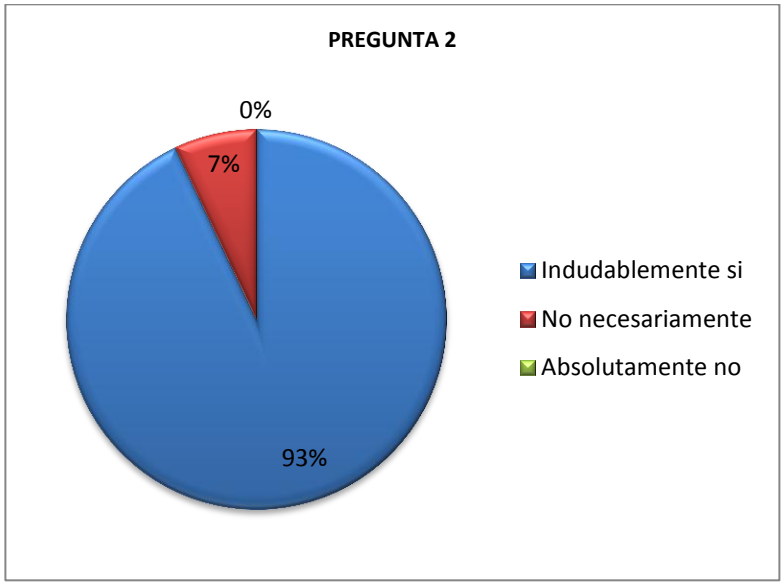

Fig. 11. Gráfica de resultados de la pregunta 2 de la encuesta.

- Interpretación.

Evidentemente en cuanto a ésta pregunta de manera rotunda el 93% de los encuestados, creen que se cumple el supuesto de que con el proceso propuesto será factible la toma de métricas que ayuden a madurar el proceso en materia de seguridad.

En cuanto al pequeño porcentaje (7%) que creé que no necesariamente se cumpliría, hace mención sobre el hecho de que en TSPi tampoco se consideran de manera clara a otros requerimientos no funcionales a parte de la seguridad.

Pregunta 3.

¿Piensas qué la propuesta presentada es factible para ser aplicada por la academia en los cursos regulares de manera satisfactoria?"

- Gráfico de porcentajes.

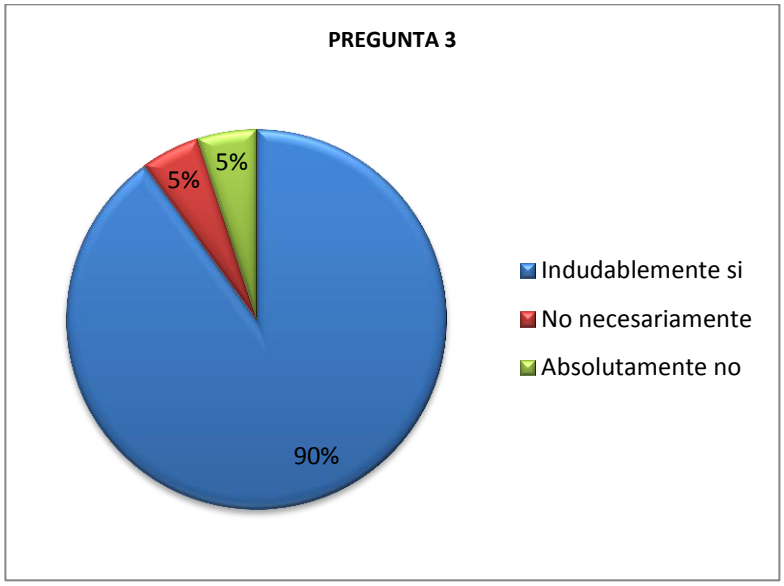

Fig. 12. Gráfica de resultados de la pregunta 3 de la encuesta.

- Interpretación.

En el caso de la pregunta tres como se puede ver, el 90% está convencido de que indudablemente la propuesta de éste trabajo facilita el aprendizaje del proceso de desarrollo y prácticas de seguridad en la academia.

En cuanto al porcentaje que cree que no necesariamente se cumple el supuesto, lo hace más bien en sentido de cuestionar si pudieran existir otras propuestas a parte de la presentada.

Pregunta 4.

¿Piensas qué la propuesta presentada es lo suficientemente flexible para ser adaptada con la gran mayoría de los ciclos de vida de desarrollo?"

- Gráfico de porcentajes.

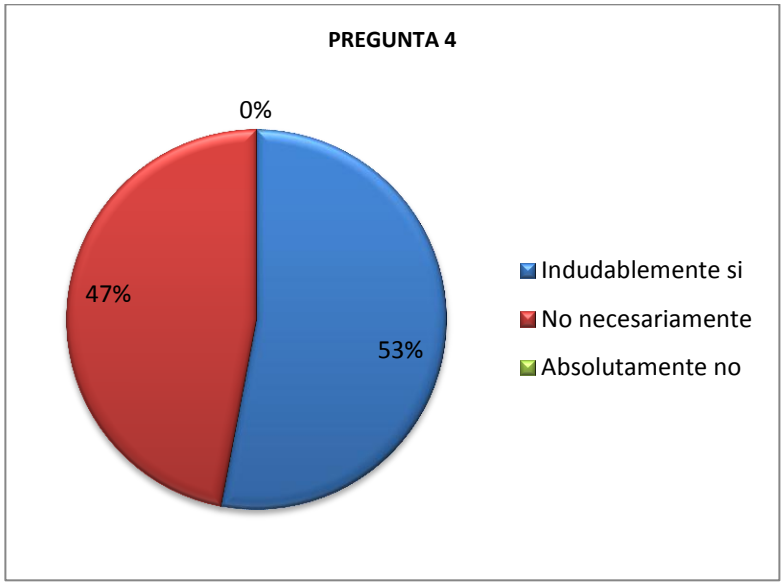

Fig. 13. Gráfica de resultados de la pregunta 4 de la encuesta.

- Interpretación.

El supuesto que se cuestiona en ésta pregunta asevera sobre la flexibilidad que la propuesta de éste trabajo tendría de ajustarse los ciclos de vida para el desarrollo de software existentes. En donde el 53% cree que si, sin embargo el porcentaje que difiere de ello lo hace argumentando que, se debe considerar qué el tamaño del software a ser desarrollado influiría sobre el ciclo de vida que se empleara y por lo tanto el comportamiento en cierta medida podría ser inesperado.

Pregunta 5.

¿Consideras qué el incluir aspectos de seguridad durante el desarrollo de la estrategia del proyecto ayude a disminuir el error de estimación de los artefactos producidos?"

- Gráfico de porcentajes.

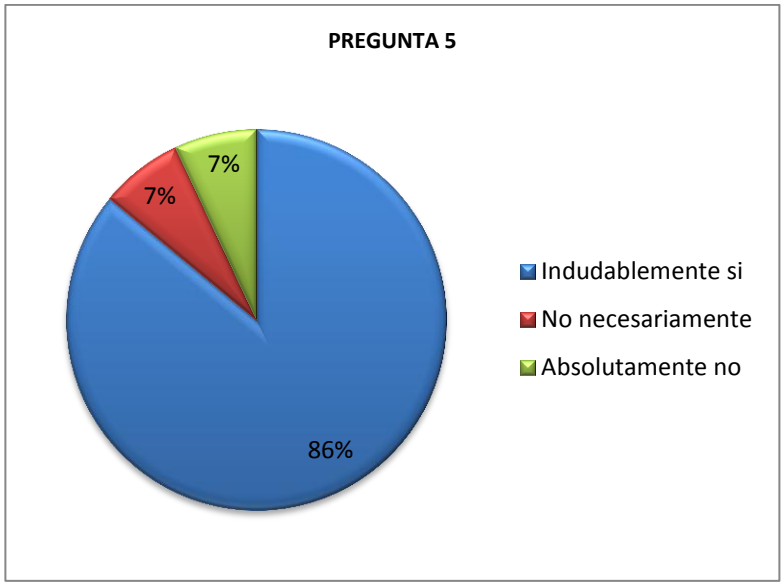

Fig. 14. Gráfica de resultados de la pregunta 5 de la encuesta.

- Interpretación.

Sin duda como se puede observar, el supuesto que se discute en ésta pregunta resulto afirmativo con un 86%, significando que efectivamente se debe contemplar a la seguridad como un aspecto que influye sobre el esfuerzo al momento de desarrollar los componentes de software. Por otro lado en cuanto al pequeño porcentaje que dice que no, lo hace sugiriendo que existen otros requerimientos no funcionales que también deben ser tomados en cuenta.

Pregunta 6.

"¿Crees qué los roles establecidos en TSPi son suficientes con respecto a la carga de trabajo que representa la inclusión de prácticas de seguridad al proceso TSPi?

- Gráfico de porcentajes.

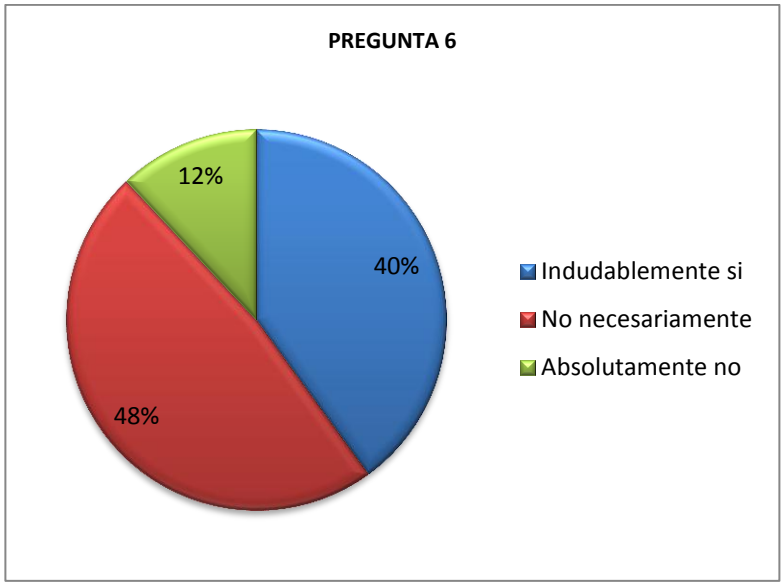

Fig. 15. Gráfica de resultados de la pregunta 6 de la encuesta.

- Interpretación.

En este caso según la apreciación de los encuestados la balanza se carga ligeramente de lado de los que creen que el supuesto presentado no necesariamente se cumple con respecto a los que creen que si (48% contra 40%). Partiendo de los argumentos de los encuestados, significaría que para proyectos pequeños los roles existentes TSPi serían suficientes y que la carga de trabajo se pudiera distribuir entre estos, pero que para proyectos grandes, se debería de pensar en agregar roles específicos para aspectos de seguridad. En tanto que un 12% está completamente convencido de que hacen falta más roles para el proceso propuesto.

6 Conclusiones y trabajo futuro.

En esta sección se escriben, en base a la evaluación realizada, las conclusiones que se obtuvieron sobre los supuestos descritos anteriormente sobre la propuesta presentada, que como ya se ha dicho trata sobre la incorporación de prácticas de seguridad en TSPi.

Así mismo se comentan las acciones futuras al respecto del presente trabajo, que darán pie a mejoras en la propuesta.

6.1 Conclusiones

PRIMERA

El proceso creado es aún un primer esfuerzo y por ende debe ser mejorado en su documentación y fortalecido en su definición. Esto es, que cuente con un marco de trabajo que contenga los scripts, estándares y formatos necesarios para que su uso sea más claro y sencillo de seguir y sus ventajas sean más evidentes.

SEGUNDA

Se concluye que el proceso propuesto en este trabajo resulta factible para la toma de métricas que resulten útiles para madurar el proceso de desarrollo de software en materia de seguridad. Sin embargo se ha detectado la necesidad de proponer métricas ya predeterminadas para cuestiones de efectividad en la implementación de la seguridad, en las cuales ya se tenga clara su utilidad y empleo.

TERCERA

Definitivamente se puede concluir que, la propuesta de este trabajo es viable para ser usada en las Universidades para la enseñanza de procesos en donde se involucren cuestiones de seguridad en el desarrollo de software. No obstante se debe considerar el poner en práctica el proceso en proyectos piloto para poder afirmarlo con datos cuantitativos.

CUARTA

De acuerdo a la evaluación realizada sobre el proceso propuesto, es posible concluir que éste es lo suficientemente flexible para con los ciclos de vida de desarrollo de software. Sin embargo, para reforzar la mencionada aseveración, será necesario poner en práctica el proceso en proyectos en los que se involucren distintos ciclos de vida e ir tomando notas sobre su comportamiento.

QUINTA

En base a la evaluación realizada se puede afirmar que, al incluir en los componentes a desarrollar actividades referentes a la seguridad, al estimar el esfuerzo requerido para desarrollarlos se tendrá un error de estimación menor. Sin embargo, será necesario aplicar el proceso en proyectos piloto en los que se tomen métricas cuantitativas en este sentido.

SEXTA

Se ha concluido en base a la evaluación realizada que, los roles del proceso TSPi no serán suficientes para soportar la carga de trabajo del proceso propuesto. Sobre todo para proyectos grandes en los que se deberá contar con un rol especialmente diseñado para llevar a cabo la administración de la seguridad, durante el proceso de desarrollo de software.

6.2 Trabajo futuro.

- Se debe revisar la definición del proceso propuesto . Esto es, tener una estructura que cuente una base robusta de scripts, estándares y los formatos necesarios para que su uso sea más claro y sencillo de seguir.
- Crear scripts para roles unificados de TSPi más CLASP y considerar un rol dedicado especialmente para la administración de tareas de seguridad en el desarrollo de software.

- Se planea realizar evaluaciones sobre la ejecución del proceso sobre un proyecto piloto. Esto a manera de caso de estudio para probar la propuesta y mejorarla.
- Medir la mejora de las estimaciones a través de proyectos que involucran el proceso propuesto en este trabajo, con respecto a los que no lo usan.
- Probar el proceso propuesto en proyectos con ciclos de vida distintos y tomar notas de su ejecución.
- Se propondrán métricas ya predeterminadas para cuestiones de efectividad en la implementación de la seguridad, para las cuales ya se tenga clara su utilidad y empleo.
- Adecuar los scripts propuestos para que el proceso pueda funcionar para proyectos de n ciclos.

Referencias.

[1] Humphrey, W. S. *Introduction to the Team Software Process .* Addison Wesley.(2000)

[2] Humphrey, W. S. *Instructor's Guide for Introduction to the Team Software Process.* Carnegie Mellon University. (2006).

[3] Humphrey, W. S., Chick, T. A., Nichols, W., & Pomeroy-Huff, M. *Team Software Process - Body of Knowledge.* SEI/CMU. (2006).

[4] Humphrey, W. S. *A Discipline for Software Engineering.* Addison Wesley. (1995).

[5] Humphrey, W. S. *Winning with Software.* Addison Wesley. (2002).

[6] Gómez V.A. *Enciclopedia de la Seguridad Informática.* Alfaomega.(2007)

[7] Secure Software, I. *CLASP - Comprehensive Lightweight Application Security Process Version 2.0.* (2006).

[8] Chandra, P. Software Assurance Maturity Model: A guide to building security into software development Version 1.0. OWASP. (2010).

[9] Microsoft. Microsoft Security Development Lifecycle 5.0. Microsoft. (2010).

[10] Win, B. D. On the secure software development process: CLASP, SDL and Touchpoints compared. *Science Direct*. (2008).

[11] *Touchpoints*. http://www.swsec.com/resources/touchpoints/

[12] *SEI-TSPi*. http://www.sei.cmu.edu/tsp/tools/tspi/index.cfm

[13] *SEI* http://www.sei.cmu.edu/

[14] *SEI-TSP* http://www.sei.cmu.edu/tsp/

[15] *Touchpoints* http://www.drdobbs.com/184415391

Anexos.

ANEXO 1.

TSPi CYCLE1 TEAM LAUNCH: SCRIPT LAU1+C

Purpose	**To start the teams on the first development cycle.**
Entry Criteria	• All the students have satisfactorily completed a PSP course • The students have read textbook Chapters 1,2,3, and Appendix A • The students have read textbook CLASP Concepts View and Role-Based View
General	This launch script starts the team projects. The principal objectives are to describe the course. • Form the teams and assign team roles. • Explain the objectives for the product to be developed. • Establish team meeting and reporting times. Steps 1, 2, and 3 are completed during the first class session. Steps 4 through 8 are completed during the second class session.

Step	Activities	Description
1	Course Overview	The instructor describes the TSPi team course objectives. • What the students are expected to accomplish • How their work will be evaluated and graded • The basic principles of teamwork • The TSPi + CLASP process
2	Student Information	The instructor explains the criteria for making team assignments. • The information needed to make proper assignments • The team roles, responsibilities, and qualifications • Complete and return the INFO form before the end of the class • Read textbook Chapter 4 an Appendix B • Read the textbook chapters on the roles that interest them • Read the CLASP Role-Based View
3	Product Objectives	The instructor describes the product objectives.

		• The critical product objectives that must be satisfied (functional and security) • The optional and desirable objectives • The criteria for evaluating the finished product
4	Team Assignments	The instructor gives the students their team and role assignments.
5	Team Goals	The instructor describes goal settings. • Why goals are needed and typical team and role goals • The instructor ensures that students engage in awareness of roles over responsibilities for security.
6	Team Meetings	The instructor explains the team meeting, its purpose, and conduct. • The meeting purpose, scheduling, and reporting • Weekly data requirements
7	The First Team Meeting	The team leader holds the first meeting of his or her team. • Discusses team members' roles • Discusses and agrees on cycle goals • Establishes a standard time for the weekly team meeting • Agrees on a specific time each week when all team members will provide their weekly data to the planning manager
8	Data Requirements	The planning manager reviews for the team the • Data required from every team member every week • Reports to be generated and provided the team from these data
9	Project Start	The team starts work on the project using the STRAT1 script.
Exit Criteria		• Each student has completed and submitted an INFO form • The development teams are formed and roles assigned. • The instructor has described the overall product objectives. • The instructor has reviewed and discussed the TSPi and the team's and role goals. • The team has agreed on cycle 1 goals, weekly meeting times, and the weekly data to report.

TSPi - CYCLE 1 Development Strategy Script: STRAT1+C

Purpose	**To guide a team through producing a TSPi development strategy and preliminary size and time estimates**
Entry Criteria	• The students have read textbook Chapter 4. • The instructor has reviewed and discussed the TSPi process. • The instructor has described the overall product objectives. • Development teams have been formed and roles assigned. • The teams have agreed on goals for their work. • The students have read the textbook CLASP Role-Based View and Activity-Assessment View.
General	The development strategy specifies the order in which product functions are defined, designed, implemented, and tested. • The way the product will be enhanced in future cycles • How to divide the development work among the team members The development strategy is produced at the beginning of the process to guide size estimating and resource planning. • If the development strategy changes during planning, requirements, or development, it must be updated. The preliminary size and time estimates • Cover the planned work for each development cycle • Provide the basis for allocating work among team members

Step	Activities	Description
1	Strategy Overview	The instructor describes the development strategy. • What the strategy is, how it is produced, and how it is used • Criteria for an effective strategy • The need for and ways to produce the size and time estimates

2	Establish Strategy Criteria	• The development manager leads discussion of strategy criteria • The meeting reporter (quality/process manager) documents these criteria and provides copies to the team members and instructor
3	Produce the Conceptual Desgin	The development manager leads the team in producing the conceptual design for the overall product. Consider security objectives. They must be represented in the Conceptual design diagram.
4	Make the Activity-Assessment for security	The Development Manager leads the team in selecting a subset of security activities of the 24 options proposed by CLASP. If the team considers it is necessary to add extra activities, it is the best time to do so.
5	Select the Development Strategy	The development manager leads the team through producing the development strategy. This involves • Proposing and evaluating alternative strategies • Allocating product functions and security requirements to each development cycle • Defining how to subdivide and later integrate the productgbr
6	Produce the Preliminary Estimate	The planning manager leads the team through producing the preliminary size and time estimates, which must include • Size and time estimates for all the current-cycle products • Rough estimates for the products of subsequent cycles
7	Assess Risks	Identify and asses project risks and enter them in ITL. Include also aspects about of security risks.
8	Document the Strategy	The meeting reporter documents the selected strategy (STRAT)
9	Produce the Configuration Management Plan	The support manager produces the configuration management plan.

		• Identifies the configuration control board and its procedures • Specifies any needed support tools and facilities • Reviews the procedures with the team for their agreement
Exit Criteria	• A completed and documented development strategy • Completed and documented size and time estimates for all product elements to be produced during the next cycle • Completed and documented estimates for the products to be produced in subsequent development cycles • Documented configuration management procedure • Risks and issues entered in the ITL log • Conceptual design and completed STRAT form • Updated project notebook	

TSPi CYCLE 1 Development Plan: Script PLAN1+C

Purpose	**To guide a team through producing individual and team task, schedule, and quality plans for development cycle 1**
Entry Criteria	• The team has a development strategy and conceptual design. • The students have read textbook Chapter 5. • The students have read the textbook CLASP Role-Based View and Activity-Assessment View.
General	The task plan defines the • Time required to perform each process task • Rough order in which the tasks will be performed • Planned value of each task The schedule plan gives • Each engineer's planned time for each project week • The total planned team hours by week • The anticipated completion week for each task • The planned value for each week If the task and schedule plans indicate the project will not be completed on time, readjust the strategy and replan.

Step	**Activities**	**Description**
1	Planning Overview	The instructor describes the planning process.

		• The task and schedule plans and how they are produced • The quality plan and how it is produced
2	Enter the Size estimates in form STRAT	Starting with the conceptual design and STRAT form produced in the strategy phase, the planning manager leads the team in • Identifying any other products to be produced and their sizes • Recording the STRAT form and other size data in SUMS
3	Produce the Task Plan	The planning manager leads the team through • Producing a task list with team and engineer time estimates • Entering these data in the TASK form
4	Produce the Schedule Plan	The planning manager obtains the estimated number of hours each team member plans to spend on the project each week and • Enters the weekly hours in the SCHEDULE form • Produces the team TASK and SCHEDULE forms • Reworks the plan if the hours are inadequate
5	Prouce the Quality Plan	The quality/process manager leads the team through • Reviewing the team's quality objectives (consider the security issues). • Estimating the defects injected and defect-removal yields • Propose metrics and set goals for these metrics. • Generating and assessing trial SUMP and SUMQ plans • Making needed process adjustments to get a satisfactory plan
6	Produce the Individual Engineering Plans	The planning manager helps the engineers make personal plans. • Allocating the tasks among team members • Estimating the time to perform each task

		• Entering the data in the TASK and SCHEDULE forms • Producing the planned-value schedule and task completion dates
7	Balance Team Workload	The planning manager leas the team through • Identifying workload imbalances • Reallocating tasks to minimize the schedule • Producing balanced engineer plans • Producing the consolidation team plan (TASK, SCHEDULE, SUMP, and SUMQ forms)
Exit Criteria		• Completed team and engineer TASK and SCHEDULE forms • Compelted SUMP, SUMQ, and SUMS forms • Updated project notebook

TSP Cycle 1 Requirements Development: Script REQ1 + C

Purpose	**To guide a team through developing and inspecting the requirements for cycle 1 of a team development project**
Entry Criteria	• The team has a development strategy and plan. • The students have read chapter 6, the test sections of Chapter 9, and the need statement. • The students have read the CLASP Activity-Assessment View and CLASP Activity-Implementation View
General	The requirements development process produces the Software Requirements Specification (SRS), which defines • the functions the product is to perform • use-case descriptions for each normal and abnormal function • the security requirements to be meet by the system The team should be cautious about expanding the requirements. • Without experience with similar applications, seemingly simple functions can take substantially more work than expected. • It is generally wise to add functions in small increments. • If more time remains, add further increments.

Step	**Activities**	**Description**
1	Requirements Process Overview	The instructor describes the requirements process and its products.

		• How the requirements process is performed • How the requirements inspection is conducted and reported
2	Need Statement Review	The development manager leads the team in reviewing the product need statement and formulating questions for the instructor about • The functions to be performed by the various product versions • How these functions are to be used
3	Need Statement Clarification	The development manager provides • consolidated questions to the instructor, who discusses the answers with the team. • The development manager provides • consolidated questions about the operating environment for the safety requirements.
4	Document security-relevant requirements	The development manager helps to • identify the functional requirements and business security issue. • provide basic business requirements for security products.
5	Requirements Tasks	The development manager leads the team through • Outlining the SRS document and the work to produce it and including documentation for the functional requirements and business security issues
5	Task Allocation	The team leader helps allocate the tasks among the team members and • Obtains commitments for when they will complete these tasks
6	Requirements Documentation	Each team member • Produces and reviews his or her portions of the SRS

		document • Provides these to the development manager The development manager produces the SRS draft.
7	System Test Plan	The development manager leads the team in producing and reviewing the system test plan (see Chapter 9 on system test).
8	Requiremnts and System Test Plan Inspection	The quality/process manager leads the team through • Inspecting the SRS draft and system test plan (see script INS) • Identifying questions and problems • Defining who will resolve each question and problem and when • Documenting the inspection in form INS
9	Requirements Update	The development manager obtains the updated SRS sections and • Combines them into a final SRS • Verifies traceability to the need statement or other sources
10	User SRS Review	• The development manager provides a copy of the final SRS to the instructor (user) for approval • After approval, the team fixes any identified problems.
11	Requirements Baseline	• The support manager baselines the SRS
Exit Criteria		• A completed and inspected SRS document and system test plan • A completed INS form for the requirements inspection • Time, defect, and size data entered in the TSPi support system • Updated project notebook

TSPi Cycle 1 Design: Script DES1 + C

Purpose	To guide a team through developing and inspecting the software design specifications for a team development project
Entry Criteria	• A development strategy and plan • A completed and inspected SRS • The students have read textbook Chapter 7 • The students have read the CLASP Activity-Implementation View

	and Vulnerability View
General	The design process produces the software design specification (SDS), which defines the overall product structure for cycle 1. • Major product components and their interface specifications • The allocation of use cases to components The SDS also specifies • File and message standard, definitions, naming convetions • Design notation and standards

Step	**Activities**	**Description**
1	Design Process Review	The instructor describes the design process an its products. • How the design proces is performed and a sample SDS • How the design inspection is conducted and reported • Design standards and convetions
2	High-Level Design	The development manager leads the team through • Defining the cycle-1 product structure • Naming the product components • Allocating use cases to these components • Apply security principles to the system design • Providing a structured basis for the security requirements of a system. • Defining the roles and capabilities / resources of the system to which the role can access. • Identifying the design tasks to be completed and documented.
3	Design Standards	The quality/process manager leads the effort to produce the name glossary and design standards.
4	Design Tasks	The developmoent manager leads the team through • Outlining the SDS document and the work to produce it
5	task Allocation	The team leader helps allocate the tasks among the team members and • Obtains commitments for when they will complete

		these tasks
6	The Design Specification	Each team member • Produces and review his or her portions of the SDS document • Provides these to the development manager The development manager produces a composite SDS draft
7	Integration Test Plan	The development manager leads the team in producing and reviewing the integration test plan.
8	Design and Integration Test Plan Inspection	The quality/process manager leads the team through inspecting the SDS draft and integratino test plan (see script INS) so that • Every use case is covered and referenced in the design • The design is complete and correct • The integration test plan is adeuquate • Each problem is recorded and fix responsibility assigned The inspection is documented in form INS, and defects are recorded in LOGD.
9	Design Update	The developmoent manager obtains the updated SDS sections and • Combines them into afinal SDS • Verifies traceability to the SRS
10	Update Baseline	The support manager baselines the SDS
Exit Criteria	• A completed and inspected SDS and integration test plan • The design standards and name glossary • Updated SUMP and SUMQ forms and INS inspection forms • Updated project notebook	

TSPi Cycle 1 Implementation: Script IMP1+C

Purpose	**To guide a team through implementing and inspecting the software for cycle 1 of a team development project**
Entry Criteria	• The team has the developoment strategy and plan. • SRS and SDS specifications and name glossary • Documented coding and other standards

	• The students have read textbook Chapter 8 • The students have read the CLASP Activity-Implementation View and Vulnerability View
General	The implementation process produces a reviewed, inspected, and unit-tested product that must • Completely cover the SDS and SRS functions and use cases • Conform to established coding and design standards • Follow the PSP2.1 or PSP3 process

Step	**Activities**	**Description**
1	Implementation Process Overview	The instructor describes the implementation process, including • The importance of a quality implementation • The importance of the security implementation • The need for and content of the coding standards • The strategy for handling poor-quality components
2	Implementation Planning	The development manager leads the work to • Define and plan the implementation tasks (SUMP, SUMQ)
3	Task Allocation	The team leader helps allocate the tasks among the team members and • Obtains commitments for when they will complete these tasks
4	Detailed Design	The engineers produce the detailed design • Do a design review using thorough design review methods • Complete forms LOGD and LOGT
5	Unit Test plan	The engineers produce the unit test plans including security test (don't forget it)
6	Test Development	The engineers follow script UT to develop the unit test cases, test procedures, and test data • Provide input validation semantic unit level. • Reliably identify errors in a structured manner at the earliest.
7	Detailed-Design Inspection	The quality/process manager leads the team in a DLD inspection of each component (script INS and forms INS and LOGD).

8	Code	The engineers produce the component source code. • Do a code review using a personal checklist. • Compile and fix the code until it compiles without error. • Complete forms LOGD and LOGT.
9	Code Inspection	The quality/process manager leads the team in a code inspection of each component (script INS and forms INS and LOGD).
10	Unit Test	The engineers, following script UT, • Conduct the unit tests and complete forms LOGD and LOGT
11	Component Quality Review	The quality/process manager reviews each component's data to determine if component quliaty meets established team criteria. • If so, the component is accepted for integration testing. • If not, the quality/process manager recommends either • That the product be reinspected and reworkds • That it be scrapped and redeveloped
12	Component Release	• When the compnents are satisfactorily implemented and inspected, the engineers release them to the support manager. • The support manager enters the components in the configuration management system.

Exit Criteria	• Completed, inspected, conifguration-controlled components • Completed INS forms for the design and code inspections • Unit test plans and support materials • Updated SUMP, SUMQ, SUMS, LOGD, and LOGT forms • Updated project notebook

TSPi Cycle-1 Integration and System Test: Script TEST1 +C

Purpose	**To guide a team through integrating and testing the product components into a working cycle-1 system**
Entry Criteria	The team has a development strategy and plan. • Completed and inspected SRS and SDS specifications

	• Implemented, inspected, and unit-tested components under configuration control The students have read textbook Chapter 9 The students have read the CLASP Activity-Implementation View and Vulnerability View
General	When defects are found in build, integration, or system test, the quality/process manager determines whether testing should continue. Every defect found in integration or system testing is recorded in the defect log (LOGD) and reviewed by the entire team to determine • Where similar defects might remain in the product • How and when to find and fix these defects • The process changes to prevent similar defects in the future

Step	**Activities**	**Description**
1	Test Process Overview	The instructor describes the integration and system test process. • The need for quality components before testing • The need for and content of testing standards • The strategy for handling poor-quality components
2	Test Development	The development manager or alternate leads test development. The team leader helps allocate the test development and testing tasks among the team members. The test team members perform their test development tasks. • Define any required build processes and procedures. • Develop the integration test procedures and facilities. • Develop the system test procedures and facilities including security aspects. • Measure the size and running time for each test. • Review the test materials and correct errors.
3	Build	The team builds the product and checks it for completeness. • Verify that all needed parts are on hand. • Build the product and provide it to integration test. • Product owner (developer) records all defects in the defect log (LOGD).
4	Integration	The development manager or alternate leads the integration tasks. • Check completeness and integration-test the product.

		• Record all test activities in the test log (LOGTEST). • Product owner (developer) records all defects in the defect log (LOGD).
5	System Test	The development manager or alternate leads the ystem test tasks. • Test the product or normal and stress conditions. • Test the product for installation, conversion, and recovery. • Record all test activities in the test log (form LOGTEST). • Product owner (developer) records all defects in the defect log (LOGD).
6	Documentation	The development manager or alternate leads the team in • Producing the user-documentation outline and tasks • Allocating these tasks to the documentation team • Reviewing the outline with the test team for completeness • Drafting the first-cycle user documentation • Reviewing, correcting, and producing the user documentation • Provide stakeholders with information about safety measures that can improve product safety. • Provide documentation for the use of functional safety of the product.
Exit Criteria	• An integrated and test cycle-1 product • Completed LOGD and LOGTEST forms for all the tests • Completed and reviewed user documentation • Time, size, and defect data entered in the TSPi support system	

TSPi - Cycle 1 Postmortem: Script PM1 + C

Purpose	• **To gather, analyze, and record project data** • To evaluate the team's and each role's performance • To identify ways to improve the cycle-2 process including security aspects. • To produce the cycle-1 report
Entry Criteria	• The engineers have completed and test the product. • They have gathered all the data and completed all the forms • The students have read textbook Chapters 10, 16, 17, and 18. • The students have read the CLASP Activity-Implementation View

	and Vulnerability View
General	The cycle-1 report contains an analysis of the project by each role. • Overall team performance: team leader • Plan versus actual performance: planning manager • Overall product design and standards: development manager • Change management and project support: support manager • Process and product quality: quality/process manager • Check the current state of security The cycle report should • Use process data to support the engineers' statements • Thoughtfully consider the meaning of the results produced • Be short and concise

Step	**Activities**	**Description**
1	Postmortem Process Overview	The instructor describes the postmortem process • The need for complete and accurate process data • The contents of the cycle report • The peer evaluation process and forms
2	Review Process Data	The quality/process manager leads the team in analyzing project data and identifying problem and improvement areas. • Leadership, planning, process, quality, or support • Suggested team actions and responsibilties • Areas for instructor or facility improvement The engineers prepare and submit PIPs on these improvement suggestions.
3	Evaluate Role Performance	The team leader leads the team in evaluating the effectiveness of the team roles, the instructor's actions, and the support facilities. • Where they were effective • Where there is room for improvement
4	Prepare Cycle-1 Report	• The team leader leads the team in outlining the cycle-1 report. • Allocating report work to the team members • Obtaining commitments for report section completion • Assembling, reviewing, and correcting the completed

		report
5	Prepare Role Evaluations	Each engineer completes an evaluation of the team and of each team role using form PEER. • Each role's difficulty and contribution • Percents must total 100% • The effectiveness of each role on a scale of 1 (inadequate) to 5 (superior).
Exit Criteria		• The development cycle has produced a high-quality product with all required documentation. • The completed product is under configuration control. • ALl process data have been evaluated and PIPs submitted. • The peer evaluations are done and submitted (PEER). • The cycle-1 report has been completed and submitted. • SUMP an SUMQ forms are completed for the system and all its components. • The project notebook has been updated.