Universidad de las Ciencias Informáticas

Facultad 4 y Laboratorio de Soluciones e Investigaciones Avanzadas en Gestión de Proyectos

Propuesta de solución de la Vista de Seguridad de la Arquitectura del Sistema GESPRO 12.05

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Dayana Cabrera Pérez.

Tutor: Ing. Yusdel Meriño Almaguer.

La Habana, Cuba, Julio del 2012

El camino siempre será difícil y requerirá el esfuerzo inteligente de todos. [...] Ser tan prudentes en el éxito como firmes en la adversidad es un principio que no puede olvidarse. [...]

Fidel Castro Ruz.
La Habana, 18 de febrero de 2008.

Declaración de autoría

Por este medio declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso que estimen pertinente con él. Para que así conste firmo la presente a los __ días del mes de __ del año __.

Dayana Cabrera Pérez	Ing. Yusdel Meriño Almaguer
_____	_____
Firma del autor	Firma del tutor

Agradecimientos

A mi familia por su apoyo incondicional a lo largo de estos cinco años.

A mis padres por toda su dedicación, comprensión, sacrificio y esmero, por confiar tanto en mí y por tan sabios consejos a lo largo de mi carrera. Papi hoy tu sueño sí se cumple completo, te lo prometí y hoy te lo estoy haciendo realidad. Mami ya ves que tanto sacrificio sí valió la pena, hoy tienes a tu hija, como tanto quisiste, hecha una profesional. Gracias por guiarme por un buen camino, por el ejemplo y educación que me dieron. Los amo mucho y me siento muy orgullosa de ustedes.

A mi hija, por darme las fuerzas y permitirme llegar hasta aquí.

A mis hermanos que han estado siempre conmigo y que siempre estarán presentes, apoyándome, en cada paso que dé en la vida.

Al papá de mi niña por darme mi mayor tesoro.

A mis sobrinas y sobrino que tanto adoro, Rosme, Sadia, Emily, Liz Marian, Claudia y José Ramón, a los cuales adoro y espero que este título les sirva de inspiración.

A mi suegra Tania, por haber cuidado tan bien de mi niña.

A mis amigos de la universidad, que me han hecho vivir innumerables momentos de alegría, risas, porque cada uno de ellos es parte de un grupo único en el que no existen misiones imposibles, gracias por hacerme la vida fácil Leannet, Darlenis, Danir y Melvin.

A unas personas muy importantes en mi vida, que sin ellos mis recuerdos en la universidad no serían tan bellos hoy, a los amigos de los años, los que no se destiñen aunque el tiempo y la distancia nos separen Patricia, Ernesto, Gretel, Dariela, doy gracias a la vida por permitirme tener tan buenos amigos como ustedes.

A mis compañeros de aula que tanto me ayudaron en mi último año.

A mi Decano José Luis Basulto que tanto me ayudó y gracias a él me gradúo hoy.

A Félix, por dedicarme todo el tiempo que necesité para terminar esta tesis.

A mi tutor Yusdel, que más que tutor ha sido un amigo incondicional en mis 5 años de carrera, gracias por todo.

A los profesores que tanto me ayudaron en mis últimos años, Yisel, Deyler, Yunesti.

A todas las personas que me consideraron y que sin esperar nada de mí de una forma u otra siempre me ayudaron.

Dedicatoria

A mi madre, por educarme y guiarme siempre por el camino correcto, a quien amo y admiro más que a nadie en la vida...

A mi mayor tesoro, mi bebé Jade, quien fue mi principal fuente de inspiración para culminar mi carrera.

Resumen

Con el auge de las tecnologías, las organizaciones cubanas han incrementado el uso de aplicaciones informáticas, con el objetivo de controlar eficientemente los recursos que se manejan en la misma. Entre las aplicaciones informáticas puestas en práctica se encuentran los sistemas de gestión de proyectos, los cuales son herramientas que integran y automatizan todos los procesos que se llevan a cabo en las empresas. Estos sistemas deben contar con una arquitectura de software adecuada y con una descripción de la misma que permita facilitar su entendimiento y servir de apoyo en el desarrollo del sistema. Actualmente la Universidad de las Ciencias Informáticas (UCI), se encuentra inmersa en la realización del sistema integral para la gestión de proyectos GESPRO.

Como resultado de la investigación realizada se presenta una propuesta de solución a la Vista de Seguridad de la Arquitectura de GESPRO 12.05. En la misma se definen una serie de niveles que permiten caracterizar la solución vista desde las funcionalidades requeridas para garantizar un sistema seguro. Se evidencia además a través de una validación final de la herramienta vista desde la comparación de esta con las existentes hasta el momento en la UCI, la importancia y factibilidad del uso de la misma para la gestión de todos los proyectos desarrollados en la institución.

Palabras Claves: Arquitectura de Software, factibilidad, gestión, seguridad.

Abstract

With the rise of technology, Cuban organizations have increased the use of computer applications; in order to efficiently control the resources that are managed in the same. Among the applications are implemented management systems projects, which are tools that integrate and automate all processes carried out in companies. These systems must have proper software architecture and a description of it which will facilitate their understanding and to support the development of the system. Currently the University of Informatics Sciences (ICU) is immersed in the realization of an integrated system for managing projects GESPRO.

As a result of the investigation presents a proposed solution to the Vista Security Architecture GESPRO 12.05. At the same defines a number of levels that can characterize the solution seen from the functionality required to ensure a secure system. Also evidenced by a final validation of the tool seen from comparing it with existing so far in the ICU, the importance and feasibility of using it for managing all projects within the institution.

Keywords: Software Architecture, feasibility, management, security.

Tabla de contenido

Índice de Figuras

Índice de Tablas

Introducción

En las últimas décadas con el auge de las tecnologías en el campo de la informática, los procesos de desarrollo de software son cada vez más complejos, dentro de este proceso constituye una solución importante a muchos de los problemas el desarrollo de la arquitectura de software, la misma, similar a los planos de un edificio o construcción, indican la estructura, funcionamiento e interacción entre las partes del software, además ayuda a la descomposición de la complejidad asociada a los proyectos informáticos.

El aumento de la complejidad de la arquitectura de software en el crecimiento de la informática y las tecnologías de la información en los procesos de negocios de la mayoría de las empresas del mundo, ha provocado un incremento en las exigencias de los clientes por adquirir productos más seguros que garanticen la confidencialidad, autenticidad, disponibilidad y confiabilidad de la información que se manipula. Es importante destacar que aquellas compañías que se caracterizan por desarrollar sistemas con altos índices de seguridad, han tenido mayores éxitos, ganando la confianza de los clientes y ratificándose como vanguardias en la industria de la informática.

En Cuba los avances alcanzados en los últimos años en la informatización de la sociedad, a partir del incremento de tecnologías de la información en todos los sectores y en particular de las redes informáticas y sus servicios asociados, y el impulso orientado por la dirección del país al desarrollo acelerado de programas que multipliquen dichos logros, han impulsado la adopción de medidas que garantizan un adecuado nivel de seguridad, de protección y de ordenamiento.

En el país la seguridad de las organizaciones, sistemas y redes de información están constantemente amenazadas por diversas fuentes que incluyen ataques de distintos tipo y origen; la ocurrencia de catástrofes, errores de operación y negligencias, aumentan los riesgos a que están expuestos los servicios y protocolos utilizados, así como el contenido de la información tratada en dichos sistemas, todo lo cual afecta severamente la confidencialidad, integridad y disponibilidad de la información.

En la Universidad de las Ciencias Informáticas (UCI) centro de altos estudios, basado en el concepto de universidad productiva, y que tiene entre sus objetivos contribuir a que la informática se convierta en una de las principales fuentes de ingresos al país, además de contribuir a la informatización de la sociedad cubana, se encuentra en estos momentos enfrascada en el desarrollo de una gama de sistemas de software a partir de convenios con diferentes empresas, tanto del ámbito nacional como internacional. Muchos de estos productos que se implementan, manejarán un volumen considerable de información de clientes y de manera general dentro de los procesos de negocios propios de la empresa en que serán instalados finalmente.

Actualmente en muchos de estos proyectos de desarrollo de software no existe un área especializada en garantizar la seguridad del entorno de desarrollo, de despliegue y de las aplicaciones propiamente. Por esta causa implementar la seguridad resulta trabajoso, ya que cada proyecto o institución la controla de forma diferente, invirtiendo tiempo y cuantiosos recursos humanos y materiales. Esto trae como consecuencia que no se cuente con un entorno seguro de desarrollo y puedan acceder a la información personas no autorizadas. Además, al no establecerse una política estricta en todas las fases del proyecto se pueden cometer errores que conlleven a huecos de seguridad del sistema. La universidad se encuentra inmersa además, en lograr la soberanía tecnológica de los sistemas creados en el país, para de esta manera evitar la dependencia de terceros países en la elaboración de los productos informáticos.

Desde el año 2009 existen en la red de centros de la UCI, ocho herramientas de Gestión de Proyectos: la herramienta *Team Foundation Server, GForce, Redmine, Trac, Bugzilla, DotProject, Jira y Microsoft Team System.* Se conoce además que un 42 % de los proyectos en su etapa de realización no utilizan ninguna herramienta, esta situación dificulta la aparición de una estrategia única para el aseguramiento de toda esa información sensible manejada por los proyectos. Como respuesta a esta problemática, surge la idea de crear la herramienta GESPRO, la cual gestionará la información de los proyectos de la universidad, la cual cumplirá con los parámetros de seguridad plasmados en el Expediente de mejora de la UCI.

Se identificaron las siguientes dificultades:

- La información del proyecto se encuentra en las máquinas de los desarrolladores, facilitando el libre acceso a la misma.

- No existe una herramienta que estandarice el proceso de gestión de software.
- Existen dificultades con la soberanía tecnológica y la sostenibilidad de las herramientas utilizadas.

Dada la problemática anterior, se plantea el siguiente **problema a resolver**: las insuficiencias en la definición de la arquitectura de los sistemas de gestión de proyectos de la Universidad de las Ciencias Informáticas está afectando la seguridad y la soberanía tecnológica de los sistemas.

Se define como **objeto de estudio**: la Arquitectura de Software y para dar solución al problema antes expuesto se plantea el siguiente **objetivo general**: desarrollar la propuesta de seguridad de la Arquitectura del Sistema GESPRO 12.05, a partir de las experiencias en herramientas anteriores.

Para dar cumplimiento al objetivo general, se definen los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación asociado a la arquitectura de software.
- Definir la Vista de Seguridad de la Arquitectura del Sistema GESPRO 12.05.
- Implantar la Vista de Seguridad de la Arquitectura del Sistema GESPRO 12.05.
- Validar los resultados.

Se tiene como **campo de acción** la Vista de Seguridad de la Arquitectura del Sistema GESPRO 12.05.

Para recopilar información y dar solución a las dificultades antes planteadas se usará como **tipo de investigación**:

Explicativa: pretende responder a preguntas sobre relaciones de causalidad. Miden no solo variables y sus relaciones (correlaciones) sino también nexos internos. Son los estudios más completos. Pueden ser difíciles, sobre todo, resulta complejo demostrar las relaciones de causalidad.

Como **hipótesis** se tiene que si se desarrolla la propuesta de Seguridad de la Arquitectura del Sistema GESPRO 12.05, entonces se pueden lograr una mayor seguridad en el sistema y la soberanía tecnológica.

Variables dependientes:

La Seguridad y la Soberanía Tecnológica.

La **población** a estudiar en el presente trabajo será:

Los centros de la red de producción de la UCI.

La muestra es: la red de centros de la sede central.

Para desarrollar la investigación se utilizaron los métodos siguientes:

Los métodos teóricos:

- El método Histórico-Lógico: para determinar las tendencias actuales de desarrollo de los modelos y enfoques arquitectónicos.
- El método Sistémico: para determinar los componentes y definir las relaciones entre estos.
- Análisis y Síntesis: para el procesamiento de la información y arribar a las conclusiones de la investigación, así como para precisar las características del modelo arquitectónico propuesto.

Entre los métodos empíricos:

- El método de la observación: este método se utilizó para la percepción selectiva de las restricciones y propiedades del sistema.
- El método de la entrevista: este método se basa en la investigación ya que la mayor parte de la información está en manos de los especialistas.

Con la realización e implantación de la arquitectura de la Vista de Seguridad de la Arquitectura del sistema GESPRO 12.05, se espera como **aporte práctico** mejorar y optimizar la gestión de proyectos en la Universidad de las Ciencias Informáticas,

donde se facilite el trabajo en los distintos centros de la misma, haciendo más factible el rendimiento y la eficacia de dicha herramienta.

El **diseño de investigación** es el **preexperimento** de Pre y Post prueba con un solo grupo, en la que hay al menos un punto de referencia inicial. La validez interna puede ser afectada fácilmente por la historia, la maduración, la elección de un grupo atípico, la regresión y las interacciones.

Se utilizó como instrumento para llevar a cabo la investigación:

- la entrevista.

Con la implantación de la propuesta de arquitectura de la Vista de Seguridad del Sistema GESPRO 12.05, se espera como **aporte práctico** un gran impacto económico y social, dado por el mejoramiento en la seguridad del sistema, logrando asegurar información sensible contenida en los sistemas desarrollados en el centro y la soberanía tecnológica lograda por el Laboratorio de Soluciones e Investigaciones Avanzadas en Gestión de Proyectos.

Al finalizar este trabajo los **resultados esperados** son los siguientes:

- Definición de la Vista de Seguridad de la Arquitectura del Sistema GESPRO 12.05.
- Implantación de la Vista de Seguridad de la Arquitectura del Sistema GESPRO 12.05.

El presente trabajo de diploma se estructura en tres capítulos:

- En el Capítulo 1: **Fundamentación Teórica,** es analizado el concepto de arquitectura de software y seguridad, entre otras definiciones estrechamente relacionadas con las mismas. Se tratan las principales corrientes teóricas de la arquitectura de software, además se identifican los métodos que permiten la evaluación de la arquitectura propuesta.
- En el Capítulo 2: **Propuesta Arquitectónica de la Vista de Seguridad,** se analizan las prácticas para el desarrollo de aplicaciones seguras, se identifican los requisitos que debe cumplir el sistema, se realiza un análisis de cómo se establecerá la seguridad en el sistema.

- En el Capítulo 3: **Validación práctica de la propuesta realizada,** se evalúa la Arquitectura de la Vista de Seguridad de GESPRO 12.05 propuesta y se analizan los resultados obtenidos.

Capítulo 1: Fundamentación teórica.

Introducción

En el presente capítulo se tratan los aspectos más importantes de la seguridad en el desarrollo de la arquitectura de software, para lograr un sistema robusto; la soberanía tecnológica en el sistema desarrollado, así como algunas definiciones de Arquitectura de Software, refiriéndose además a la Vista de Seguridad de la Arquitectura del Sistema GESPRO. Se abordan temas referentes a la seguridad informática, su importancia, la problemática actual de la seguridad en el software, la confidencialidad de la información, entre otros aspectos.

1.1 Definiciones de Arquitectura de Software

En la actualidad no se ha llegado a un acuerdo en cuanto a la definición formal de lo que es arquitectura de software. De acuerdo con la IEEE (*Institute of Electrical and Electronics Engineers*), una arquitectura de software es: la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente; y los principios que orientan su diseño y evolución (1).

La arquitectura de software es el conjunto de planos de un desarrollo de software. Planos con las características más importantes resaltadas dejando de lado los detalles. Características como requisitos de los usuarios e inversores, plataforma (sistema operativo, hardware, base de datos, protocolos de red), bloques de construcción reutilizables, consideraciones de implantación, sistemas heredados y requisitos no funcionales (2).

Es una vista del sistema que incluye los componentes principales del mismo, el modo de actuar, las formas en que estos interactúan entre sí y se coordinan para alcanzar el objetivo del sistema. Las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución, son fundamentales para la integración satisfactoria de los mismos.

Es claro que una arquitectura de software es mucho más que la mera disposición de componentes y engloba aspectos conceptuales que tienen que ver con decisiones de

alto nivel que se toman y que tienen impacto en la forma en que organiza el sistema. La descomposición de la arquitectura en partes especializadas que permiten un mejor control y una mayor abstracción. La descomposición divide a un sistema complejo en subsistemas, paquetes y componentes especializados que fomentan la facilidad de crecimiento, la arquitectura de software debe mostrar la forma en que tales piezas de descomposición interaccionan entre si y como se utilizan los servicios que prestan cada una de ellas sin especificar los detalles de cómo cada elemento implementa el servicio o su comportamiento (3).

Por todo ello se considera a la arquitectura de software como la columna vertebral en el desarrollo de un sistema automatizado, su correcto uso y el detallado estudio de la misma, permite llegar a un exitoso resultado en el proyecto a ejecutar.

1.2 Tendencias de la Arquitectura de Software

En la década de 1990 se establece la Arquitectura de Software como un dominio separado de la Ingeniería de Software y del diseño. Aunque no existe una bibliografía explícita referente a las escuelas de Arquitectura de Software, ni se han publicado estudios de las particularidades de cada una, en la actualidad se pueden distinguir seis corrientes:

Arquitectura como etapa de ingeniería y diseño orientada a objetos: es el modelo de James Rumbaugh, Ivar Jacobson, Grady Booch, Craig Larman y otros, ligado estrechamente al mundo de UML y Rational. No caben dudas de que se trata de una corriente específica, de lo que sí puede dudarse es que se trate de una postura arquitectónica, o sea, que la arquitectura se restringe a las fases iniciales y preliminares del proceso y concierne a los niveles más elevados de abstracción, pero no está sistemáticamente ligada al requerimiento que viene antes o a la composición del diseño que viene después, sino que importa más la abundancia, el detalle de diagramas y técnicas disponibles que la visión estructural del conjunto (6).

Lo que sigue al momento arquitectónico es *business as usual*, y a cualquier configuración, topología o morfología de las piezas del sistema se la llama arquitectura. En esta escuela, si bien se reconoce el valor primordial de la abstracción (nadie después de Dijkstra osaría oponerse a ello) y del ocultamiento de información promovido por Parnas, estos conceptos tienen que ver más con el encapsulamiento en clases y objetos que con la visión de conjunto arquitectónica. Para este movimiento, la

arquitectura se confunde también con el modelado y el diseño, los cuales constituyen los conceptos dominantes. En esta corriente se manifiesta predilección por un modelado denso y una profusión de diagramas, tendiente al modelo metodológico CMM o a UPM (6).

Cuando aquí se habla de estilos, también se confunde con patrones arquitectónicos o de diseño. Jamás se hace referencia a los lenguajes de descripción arquitectónica, que representan uno de los *assets* reconocidos de la Arquitectura de Software; sucede como si la disponibilidad de un lenguaje unificado de modelado los tornara superfluos. La definición de arquitectura que se promueve en esta corriente tiene que ver con aspectos formales a la hora del desarrollo; esta arquitectura es isomorfa a la estructura de las piezas de código. Las definiciones revelan que la Arquitectura de Software, en esta perspectiva, concierne a decisiones sobre organización, selección de elementos estructurales, comportamiento, composición y estilo arquitectónico susceptibles de ser descriptas a través de las cinco vistas clásicas del modelo 4+1 de Kruchten (18).

Fueron detectadas algunas debilidades para los objetivos trazados en la investigación de esta corriente, entre ellas se destacan las limitaciones en la especificación de elementos de alto impacto en las decisiones arquitectónicas, tales como (usabilidad e interfaz, marcos tecnológicos, peculiaridades tecnológicas de los datos, configuraciones, patrones de solución de escalabilidad y solución a escenarios críticos del desliere, patrones de diseño). Asumir los modelos o resultados formales de esta tendencia pueden hacer vulnerables, elementos de gran impacto y que constituyen riesgos de transcendencia.

Arquitectura estructural, basada en un modelo estático de estilos, ADLs y vistas: constituye la corriente fundacional y clásica de la disciplina. Los representantes de esta corriente son todos académicos, mayormente de la Universidad Carnegie Mellon en Pittsburgh: Mary Shaw, Paul Clements, David Garlan, Robert Allen, Gregory Abowd, John Ockerbloom. Se trata también de la visión de la Arquitectura de Software dominante en la academia, y aunque es la que ha hecho el esfuerzo más importante por el reconocimiento de la Arquitectura de Software como disciplina, sus categorías y herramientas son todavía mal conocidas en la práctica industrial (18).

En el interior del movimiento se pueden observar distintas divisiones. Hay una variante informal de “boxología” y una vertiente más formalista, representada por el grupo de Mark Moriconi en el SRI de Menlo Park. En principio se pueden reconocer tres

modalidades en cuanto a la formalización; los más informales utilizan descripciones verbales o diagramas de cajas, los de talante intermedio se sirven de ADLs y los más exigentes usan lenguajes formales de especificación como CHAM y Z. En toda la corriente, el diseño arquitectónico no solo es el de más alto nivel de abstracción, sino que además no tiene por qué coincidir con la configuración explícita de las aplicaciones; rara vez se encontrarán referencias a lenguajes de programación o piezas de código, y en general nadie habla de clases o de objetos. Mientras algunos participantes excluyen el modelo de datos de las incumbencias de la Arquitectura de Software (Shaw, Garlan, entre otros), otros insisten en su relevancia (Medvidovic, Taylor). Todo estructuralismo es estático; hasta fines del siglo XX, no está muy claro el tema de la posición del modelado arquitectónico en el ciclo de vida (6). La potente conceptualización de las estructuras y marcos analíticos de la arquitectura, soportados en los cánones teóricos de la disciplina, hacen de esta escuela el punto de referencia a seguir en la continuidad de esta investigación (6).

Estructuralismo arquitectónico radical: se trata de un desprendimiento de la corriente anterior, mayoritariamente europeo, que asume una actitud más confrontativa con el mundo UML. En el seno de este movimiento hay al menos dos tendencias, una que excluye de plano la relevancia del modelado orientado a objetos para la Arquitectura de Software y otra que procura definir nuevos metamodelos y estereotipos de UML como correctivos de la situación (6).

Arquitectura basada en patrones: reconoce la importancia de un modelo emanado históricamente del diseño orientado a objetos, esta corriente surgida hacia 1996 no se encuentra tan rígidamente vinculada a UML en el modelado, ni a CMM en la metodología. El texto sobre patrones que esta variante reconoce como referencia es la serie POSA de Buschmann y otros y secundariamente el texto de la Banda de los Cuatro. La diferencia entre ambos textos sagrados de la comunidad de patrones no es menor; en el primero, la expresión "*Software Architecture*" figura en el mismo título; el segundo se llama *Design Patterns: Elements of reusable Object-Oriented* software y su tratamiento de la arquitectura es mínimo. En esta manifestación de la Arquitectura de Software prevalece cierta tolerancia hacia modelos de procesos tácticos, no tan macroscópicos, y eventualmente se expresa cierta simpatía por las ideas de Martin Fowler y las premisas de la programación extrema. El diseño consiste en identificar y articular patrones preexistentes, que se definen en forma parecida a los estilos de arquitectura (6).

Arquitectura procesual: desde comienzos del siglo XXI, con centro en el SEI y con participación de algunos (no todos) los arquitectos de Carnegie Mellon de la primera generación y muchos nombres nuevos de la segunda: Rick Kazman, Len Bass, Paul Clements, Felix Bachmann, Fabio Peruzzi, Jeromy Carrière, Mario Barbacci, Charles Weinstock. Intenta establecer modelos de ciclo de vida y técnicas de elicitación de requerimientos, *brainstorming*, diseño, análisis, selección de alternativas, validación, comparación, estimación de calidad y justificación económica específicas para la arquitectura de software. Otras variantes dentro de la corriente procesual se caracterizan de otras maneras como etapas del proceso: extracción de arquitectura, generalización, reutilización (6).

En el seno de este movimiento hay al menos dos tendencias, una que excluye de plano la relevancia del modelado orientado a objetos para la arquitectura de software y otra que procura definir nuevos metamodelos y estereotipos de UML como correctivos de la situación (18).

Arquitectura basada en escenarios: es la corriente más actual. Se trata de un movimiento predominantemente europeo, con centro en Holanda. Recupera el nexo de la Arquitectura de Software con los requerimientos y la funcionalidad del sistema, ocasionalmente borroso en la arquitectura estructural clásica.

Los teóricos y practicantes de esta modalidad de arquitectura se inscriben dentro del canon delineado por la arquitectura procesual, respecto de la cual el movimiento constituye una especialización. En esta corriente suele utilizarse diagramas de casos de uso UML como herramienta informal u ocasional, dado que los casos de uso son uno de los escenarios posibles. Los casos de uso no están orientados a objeto. Los autores vinculados con esta modalidad han sido, aparte de los codificadores de ATAM, CBAM, QASAR y demás métodos del SEI, los arquitectos holandeses de la Universidad Técnica de Eindhoven, de la Universidad Brije, de la Universidad de Groningen y de Philips Research: MugurelIonita, Dieter Hammer, Henk Obbink, Hans de Bruin, Hans van Vliet, Eelke Folmer, Jilles van Gurp, Jan Bosch. La profusión de holandeses es significativa; la Universidad de Eindhoven es, incidentalmente, el lugar en el que surgió lo que P. I. Sharp proponía llamar la escuela arquitectónica de Dijkstra (6).

El estudio realizado sobre las seis corrientes principales de la Arquitectura de Software permite concluir que ninguna de estas integra de manera sistémica los elementos

teóricos de las restantes, sin embargo se puede observar que predomina como elemento común en los elementos conceptuales conductivos en el diseño y desarrollo de la Arquitectura de Software, las vistas arquitectónicas, que describen características de la arquitectura en un plano específico de la solución.

1.3 Modelos Arquitectónicos

Una vista arquitectónica debe representar un aspecto parcial de una arquitectura de software, que muestra propiedades específicas del sistema. Bass haciendo uso indistinto de los términos estructura y vista, propone que las estructuras arquitectónicas pueden definirse agrupando los componentes y conectores de acuerdo a la funcionalidad del sistema, sincronización y comunicación de procesos, distribución física, propiedades estáticas, propiedades dinámicas y propiedades de ejecución (18).

Kazma, Bass, Hofmeister y Kruchten, proponen según las características del sistema, proceso de desarrollo o metodología del mismo, distintas vistas arquitectónicas. Estas vistas propuestas no son independientes entre sí, sino perspectivas distintas de un mismo sistema. Por tal motivo, las vistas arquitectónicas deben estar coordinadas, de manera tal que al realizar cambios, estos se vean correctamente reflejados en las vistas afectadas, garantizando consistencia entre las mismas (18).

De acuerdo a la definición del estándar 1471 de la IEEE:

“Una vista es una representación de un sistema completo desde la perspectiva de un conjunto de *concerns* relacionados” (35).

David Parnas las estructura en tres grupos**:**

1 Estructura de módulos: es parte de o comparte el mismo secreto que la asignación de trabajo.

2 Estructura de uso: depende de la corrección de programas.

3 Estructura de procesos: brinda trabajo computacional a procesos.

Perry y Wolf: Reconocen la necesidad de vistas que enfatizan ciertos aspectos arquitectónicos útiles para distintas audiencias o para diferentes propósitos (37).

Siemens Corporate Research propone:

- Vista conceptual: principales elementos de diseño y su interrelación.
- Vista de módulos: estructura funcional y de capas.
- Vista de ejecución: estructura dinámica.
- Vista de código: organización de código fuente, binarios y bibliotecas en el ambiente de desarrollo (37).

Libro “Software Systems Architecture” de Nick Rozanski y Eóin Woods:

- Vista funcional.
- Vista de información.
- Vista de concurrencia.
- Vista de desarrollo.
- Vista de despliegue.
- Vista operacional (37).

Propuestas de Microsoft:

- Vista Conceptual: Es usada para definir los requerimientos funcionales y la visión que los usuarios del negocio tienen de la aplicación y describir el modelo de negocio que la arquitectura debe cubrir. Esta vista muestra los subsistemas y módulos en los que se divide la aplicación y la funcionalidad que brinda dentro de cada uno de ellos. Casos de Uso, Diagramas de Actividad, Procesos de Negocio, Entidades del Negocio, entre otros.
- Vista Lógica: Muestra los componentes principales de diseño y sus relaciones de forma independiente de los detalles técnicos y de cómo la funcionalidad será implementada en la plataforma de ejecución. Los arquitectos crean modelos de diseño de la aplicación, los cuales son vistas lógicas del modelo funcional y que describen la solución. Realización de los Casos de Uso, subsistemas, paquetes y clases de los casos de uso más significativos arquitectónicamente.

- Vista Física: Ilustra la distribución del procesamiento entre los distintos equipos que conforman la solución, incluyendo los servicios y procesos de base. Los elementos definidos en la vista física se "mapean" a componentes de software (servicios, procesos, entre otros) o de hardware que definen más precisamente cómo se ejecutará la solución.

- Vista Implementación: Describe cómo se implementan los componentes físicos mostrados en la vista de distribución agrupándolos en subsistemas organizados en capas y jerarquías, ilustra, además las dependencias entre estos. Básicamente, se describe el mapeo desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (37).

Propuestas de Arquitectura 4+1 Vistas de Kruchten (RUP):

- Vista lógica: requerimientos de comportamiento, mecanismos comunes de diseño (basada en diagramas de objetos y clases).

- Vista de procesos: distribución, integridad, tolerancia a fallas (basada en describir una red lógica de programas que se comunican).

- Vista de desarrollo o implementación: rehúso, portabilidad, asignación de requerimientos y trabajo de equipos. Organización del software en el ambiente de desarrollo.

- Vista física: disponibilidad, confiabilidad, performance, escalabilidad. Mapea elementos de las otras vistas a nodos de procesamiento.

- Vista de Casos de Uso o Escenarios: definición de procesos, agrupamiento en paquetes (37).

Ninguno de estos modelos excluye a los otros, los mismos abordan las partes constituyentes de la arquitectura de software, su totalidad y la forma en que se comparta una vez construida, o durante el proceso de construcción. Sin embargo, estos modelos descuidan el tema de la seguridad, que es el eje de la presente investigación.

Por todo lo analizado anteriormente es necesario continuar con el estudio de otros modelos arquitectónicos, debido a que en los propuestos por las principales escuelas de arquitectura, sus autores no prestan especial atención a la Vista de Seguridad.

1.4 Vista Arquitectónica

Las vistas arquitectónicas predominan en los variados marcos como factor común de elementos conceptuales conductivos en el diseño y desarrollo de la arquitectura de software. Se conceptualiza Vista Arquitectónica como abstracción formalizada en prosa, lenguaje de modelado, gráfico o esquema informal, desde la que se describe las características arquitectónicas de un plano específico de la solución según el interés de los involucrados.

En los modelos de vistas arquitectónicas consultados, se identifica la no existencia de un conjunto “canónico” de vistas. Existe diversidad de planteamientos de cómo formalizar las distintas perspectivas de un sistema, en ocasiones iguales pero identificadas con nombres diferentes y viceversa. Lo más adecuado sería seleccionarlas en función de satisfacer el interés de todos los involucrados (9).

Es por esta razón que en la universidad surge la necesidad de estructurar sus propias vistas. Se basa en las vistas como metodología para el diseño de Arquitectura de Software de sistemas de información, centrándose en el papel del proceso en la gestión de los proyectos informáticos, la definición de los productos y su evolución.

La Guía base y el modelo en general impactan positivamente en la gestión del proyecto a través de los siguientes aspectos:

- Introducción de los conceptos básicos y metas técnicas del producto y por lo tanto determinación del alcance y la complejidad del proyecto (13).
- Definición y delimitación de las responsabilidades y competencias de los roles de la disciplina.
    - Facilidades para el trabajo colaborativo y en equipo, sobre objetivos comunes (atributos) y el mismo objeto (arquitectura).
    - Facilidades para la comunicación y el intercambio de información desde las diferentes perspectivas y niveles de abstracción (13).
- Establecimiento de reglas para el manejo de la guía base como metodología, potenciando el análisis del producto de diversas perspectivas minimizando los riesgos y asegurando la calidad del producto final así como una planificación más detallada (13).

- Facilita el entendimiento del producto a desarrollar y acorta el tiempo de integración del equipo, pues garantiza un lenguaje común y la coordinación necesaria de las decisiones.
    - Establecimiento de guías metodológicas durante el desarrollo.
    - Análisis a través de diferentes perspectivas (vistas).
    - Análisis, caracterización y clasificación de problemas.
    - Suministro de patrones de solución para problemas tipo.
    - Mecanismos de autoevaluación de metas por vistas, facilitando la retroalimentación (13).
- Incorporación, almacenamiento y socialización de información y conocimiento:
    - Análisis de tendencias y regularidades para entornos colaborativos.
    - Evolución de la experiencia del modelo expresada en soluciones y problemas tipos recolectados del entorno colaborativo (13).

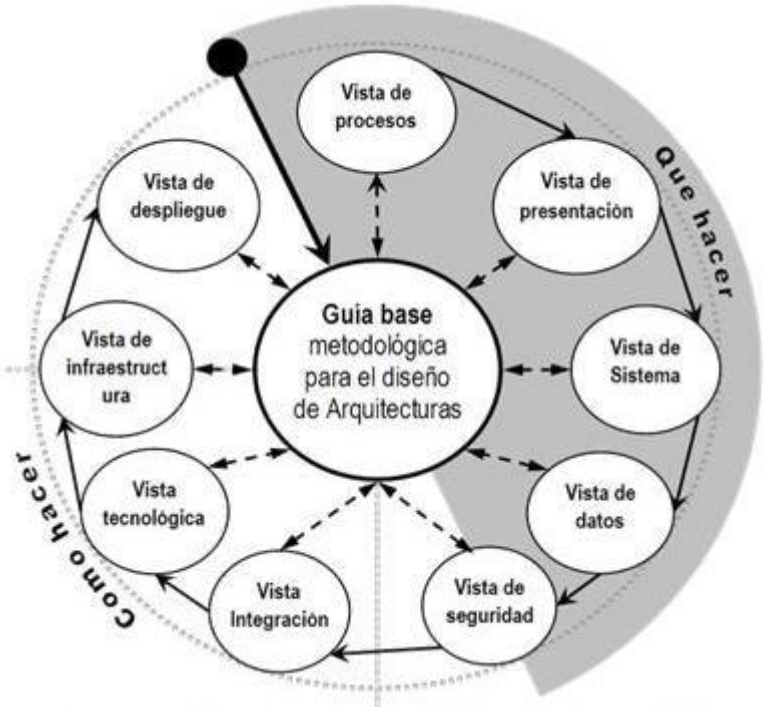


Figura 1. Guía Base de Análisis Arquitectónico

El modelo arquitectónico de referencia basado en vistas, facilita el entendimiento y análisis de la arquitectura, promueve el trabajo en grupo y da soporte al aprendizaje desde la práctica. Permite además la especialización del arquitecto según áreas concretas de actuación (13).

1.5 Vista de seguridad

Las nueve vistas de la arquitectura fueron agrupadas en tres vistas principales. La Vista de Seguridad se encuentra dentro de la Vista de Arquitectura de Tecnología.

En general la seguridad de las aplicaciones se establecerá en profundidad y por niveles, en ese sentido se presenta en esta sección lo necesario para cada uno de los niveles.

Las siguientes orientaciones constituyen buenas prácticas para el desarrollo de aplicaciones seguras:

1. Establecer un proceso de seguridad.

2. Definir las metas de seguridad del producto desarrollado.

3. Considerar la seguridad como una funcionalidad del producto.

4. Aprender de los errores cometidos.

5. Usar el principio del menor privilegio.

6. Usar la defensa en profundidad.

7. Asumir que los sistemas externos son inseguros.

1.5.1 Identificación de requisitos de seguridad del producto

Los requisitos son una etapa clave en el ciclo de vida del software, suelen ser además una combinación compleja de los requisitos de diferentes personas en diferentes niveles de una organización y del entorno en el cual operará el software. Deben ser lo más claros y no ambiguos que se pueda, y cuantificables (20).

Se especifican en este nivel las preguntas mínimas que deben quedar respondidas y que deben permitir identificar requisitos de seguridad, cada una de ellas permite establecer un nivel de seguridad, dado el rol desempeñado en el proyecto así como para las diferentes audiencias que interaccionarán con el producto. Se aborda además el entorno en que se desplegará el producto, ya sea una aplicación web publicada en Internet, una aplicación empotrada en celulares o una aplicación en red cerrada.

1.5.1.1 Aplicación web

Las aplicaciones web son soluciones informáticas que los usuarios utilizan accediendo a un servidor a través de Internet o su red interna (intranet). Como interfaz con la aplicación se utiliza un navegador de Internet (23).

Las ventajas son múltiples:

- Curva de aprendizaje rápida, el concepto de hipervínculo está muy extendido entre los usuarios.
- No existen costes de licencia.
- Basadas en arquitectura cliente/servidor.
- Los datos y el procesamiento están centralizados en el servidor (no requiere hardware adicional en las terminales).
- No hay límite en el número de terminales.
- Compatible con todos los sistemas operativos.
- Las actualizaciones son inmediatas, ya que no requieren instalación (23).

Sin embargo se observaron deficiencias, que harían el sistema en cuestión vulnerable, como son:

Los usuarios pueden acceder a la aplicación mediante un navegador de Internet, por lo que el acceso a la misma puede realizarse desde diferentes tipos de máquinas, PCs, Macs, PDAs, entre otros.

1.5.1.2 Aplicaciones empotradas en celulares

Uno de los principales motores de la sociedad de la información ha sido, sin duda, las comunicaciones móviles. Dejando a un lado la evidente influencia de Internet en la Sociedad de la Información, el cambio que ha producido en el sector de las telecomunicaciones la introducción de la movilidad ha sido crítico, no sólo por la

extensión de la posibilidad de la comunicación en cualquier momento y en cualquier lugar, sino por la propia personalización en la naturaleza de la comunicación entre individuos (24).

Una de las principales desventajas de los terminales móviles frente a la navegación en la web con PC es el reducido tamaño de las pantallas. En la web, las páginas pueden incluir diferentes alternativas de navegación (menús, enlaces en los contenidos, entre otras) en diferentes zonas de una misma página, mientras que en los dispositivos móviles la navegación se resuelve en una estructura arborescente que organiza los servicios y contenidos por áreas temáticas. Tal es el caso, por ejemplo, de la *homepage* de un portal que contenga diez enlaces temáticos; al pulsar sobre uno de ellos, se entraría en un subnivel de esa área, y así progresivamente hasta encontrar un servicio o contenido final (24).

1.5.1.3 Estructura de una red

En toda red existe una colección de máquinas para correr programas de usuario (aplicaciones). Se seguirá la terminología de una de las primeras redes, denominada ARPANET, y se llamarán hostales a las máquinas antes mencionadas. También, en algunas ocasiones se utiliza el término sistema terminal o sistema final. Los hostales están conectados mediante una subred de comunicación, o simplemente subred. El trabajo de la subred consiste en enviar mensajes entre hostales, de la misma manera como el sistema telefónico envía palabras entre la persona que habla y la que escucha (21).

1.5.1.2 Redes de área local (LAN)

Uno de los sucesos más críticos para la conexión en red lo constituye la aparición y la rápida difusión de la red de área local (LAN) como forma de normalizar las conexiones entre las máquinas que se utilizan como sistemas ofimáticos. Como su propio nombre indica, constituye una forma de interconectar una serie de equipos informáticos. A su nivel más elemental, una LAN no es más que un medio compartido (como un cable coaxial al que se conectan todas las computadoras y las impresoras) junto con una serie de reglas que rigen el acceso a dicho medio (21). Este tipo de red brinda la posibilidad de centralizar información o procedimientos, facilita la administración y la gestión de los equipos, así como una mayor seguridad de toda esta información

manejada por el sistema. Es por todo lo planteado anteriormente que se considera pertinente el uso de este tipo de red para desplegar la aplicación.

1.5.1.2.1 Características importantes

- Tecnología *broadcast* (difusión) con el medio de transmisión compartido.
- Capacidad de transmisión comprendida entre 1 Mbps y 1 Gbps.
- Extensión máxima no superior a 3 km (una FDDI puede llegar a 200 km).
- Uso de un medio de comunicación privado.
- La simplicidad del medio de transmisión que utiliza (cable coaxial, cables telefónicos y fibra óptica).
- La facilidad con que se pueden efectuar cambios en el hardware y el software.
- Gran variedad y número de dispositivos conectados.
- Posibilidad de conexión con otras redes.
- Limitante de 100 m, puede llegar a más si se usan repetidores (22).

1.5.1.3 Servicios de infraestructura de seguridad que brinda el entorno donde se usará el software

LDAP surge como una alternativa a DAP (es un protocolo a nivel de aplicación, por lo que, tanto el cliente como el servidor debían implementar completamente la torre de protocolos OSI). LDAP utiliza TCP/IP en lugar de los protocolos OSI. TCP/IP requiere menos recursos y está más disponible, especialmente en ordenadores de sobremesa. Este modelo funcional es más simple y ha eliminado opciones raramente utilizadas. LDAP es más fácil de comprender e implementar. Además representa la información mediante cadenas de caracteres en lugar de complicadas estructuras ASN.1 (25).

1.5.2 Autenticación y capa de presentación

1.5.2.1 Mecanismos seguros de encriptación y trasferencia de datos

Mantener la privacidad de los usuarios debe ser un objetivo permanente del sitio. Para ello se debe contar con una Política de Privacidad formal y explícita en el sistema y, además, deben existir mecanismos de seguridad concretos para proteger la información manejada por los mismos. Para lograr esto, uno de los mecanismos más utilizados es la incorporación de mecanismos de encriptación de los datos para información sensible, la cual consiste en transformar datos legibles en ilegibles con el objetivo de resguardar cierta información que viaja por la red. Por ejemplo, los

números de las tarjetas de crédito son encriptados para luego ser desencriptados solo por el destinatario mediante una clave especial (26).

1.5.2.2 AAA (Autenticación, Autorización, Auditoría)

En seguridad informática, el acrónimo AAA corresponde a un tipo de protocolos que realizan tres funciones: Autenticación, Autorización y Auditoría (*Authentication*, *Authorization* and *Audit*). La expresión protocolo AAA no se refiere a un protocolo en particular, sino a una familia de protocolos que ofrecen los tres servicios citados.

La Autenticación es el proceso por el que una entidad prueba su identidad ante otra. Normalmente la primera entidad es un cliente (usuario, ordenador, entre otras) y la segunda un servidor (ordenador). La Autenticación se consigue mediante la presentación de una propuesta de identidad (un nombre de usuario) y la demostración de estar en posesión de las credenciales que permiten comprobarla. Ejemplos posibles de estas credenciales son las contraseñas, los testigos de un solo uso (*one-time tokens*), los Certificados Digitales, o los números de teléfono en la identificación de llamadas. Viene al caso mencionar que los protocolos de autenticación digital modernos permiten demostrar la posesión de las credenciales requeridas sin necesidad de transmitirlas por la red (27).

La Autorización se refiere a la concesión de privilegios específicos (incluyendo "ninguno") a una entidad o usuario basándose en su identidad (autenticada), los privilegios que solicita, y el estado actual del sistema. Las autorizaciones pueden también estar basadas en restricciones, tales como restricciones horarias, sobre la localización de la entidad solicitante, la prohibición de realizar *logins* múltiples simultáneos del mismo usuario, entre otros. La mayor parte de las veces el privilegio concedido consiste en el uso de un determinado tipo de servicio. Ejemplos de tipos de servicio son: filtrado de direcciones IP, asignación de direcciones, asignación de rutas, asignación de parámetros de Calidad de Servicio, asignación de Ancho de banda, y Cifrado (27).

La Auditoría proporciona una pista de auditoría del tiempo que ha permanecido conectado cada usuario, cómo establecieron la conexión y de qué dirección IP procedían. Esto permite a los administradores revisar fácilmente problemas de seguridad y de acceso operativo que hayan tenido lugar en el pasado (27).

1.5.2.3 Auditorías de Trazas

Una traza de auditoría es un registro histórico de todos los cambios (inserciones, borrados o actualizaciones) de la base de datos, junto con la información sobre el usuario que realizó el cambio y en qué momento Obviamente, la traza de auditoría ayuda a la seguridad. Cada sistema de bases de datos realiza la traza de auditoría de forma diferente (si la tienen) (36).

1.5.3 Nivel de la capa de negocio

1.5.3.1 Tratamiento de las excepciones

Una excepción en términos de lenguaje de programación es la indicación de un problema que ocurre durante la ejecución de un programa. Sin embargo, la palabra excepción se refiere a que este problema ocurre con poca frecuencia, generalmente cuando existe algún dato o instrucción que no se apega al funcionamiento del programa por lo que se produce un error. El manejo de excepciones permite al usuario crear aplicaciones tolerantes a fallas y robustos (resistentes a errores) para controlar estas excepciones y que pueda seguir ejecutando el programa sin verse afectado por el problema (28).

1.5.3.2 Trazas del trabajo del usuario en la aplicación

En todo sistema informático llega un momento en el que surgen problemas. Ninguno es una excepción. Ante comportamientos inesperados, cuando un error inexplicable aparece, siempre se plantea la pregunta: ¿Qué está pasando? Para responder esa interrogante se hace necesario la creación de un mecanismo de registro oficial de eventos (trazas) y datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre, ya sea para un módulo del sistema en particular o en toda la aplicación (29).

Las trazas son utilizadas, entre otras cosas, para comprobar que se realizan exactamente las funciones esperadas, y no otras, así como acreditar las validaciones de datos previstas, sin modificar el sistema en ningún momento. También son usadas con el fin de monitorear las acciones que se realicen (acceso a ficheros, dispositivos, empleo de los servicios, entre otros), y para detectar indicios de hechos relevantes a los efectos de la seguridad que puedan afectar la estabilidad o el funcionamiento del sistema informático. Para ello se recurre a productos de software muy potentes y

modulares que, entre otras funciones, rastrean los caminos que siguen los datos a través del programa (29).

Todo lo planteado anteriormente garantiza un mayor nivel de seguridad en el sistema, por lo que se ha definido un factor importante en la investigación realizada.

1.5.4 Capa de datos

1.5.4.1 Inyección de código

La inyección SQL consiste en la modificación del comportamiento de las consultas realizadas en el sistema mediante la introducción de parámetros no deseados en los campos a los que tiene acceso el usuario. Este tipo de errores puede permitir a usuarios malintencionados acceder a datos a los que de otro modo no tendrían acceso y, en el peor de los casos, modificar el comportamiento de las aplicaciones (32).

1.5.5 Seguridad del Servidor de aplicación

Un cortafuego (firewall en inglés) es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Es un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar, descifrar, el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios (33).

Los cortafuegos pueden ser implementados en hardware o software, o una combinación de ambos. Los cortafuegos se utilizan con frecuencia para evitar que los usuarios de Internet no autorizados tengan acceso a redes privadas conectadas a Internet, especialmente intranets. Todos los mensajes que entren o salgan de la intranet pasan a través del cortafuego, que examina cada mensaje y bloquea aquellos que no cumplen los criterios de seguridad especificados. También es frecuente conectar al cortafuego a una tercera red, llamada Zona desmilitarizada o DMZ, en la que se ubican los servidores de la organización que deben permanecer accesibles desde la red exterior (33).

Se deben analizar todos los niveles que se desempeñan en el entorno, identificando cuál es el estado de la seguridad de la aplicación (10).

1.5.8 Monitoreo de la seguridad y proceso de desarrollo seguro

1.5.8.1 Logs

Un log es un registro oficial de eventos durante un rango de tiempo en particular. Para los profesionales en seguridad informática es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación (38).

La mayoría de los logs son almacenados o desplegados en el formato estándar, el cual es un conjunto de caracteres para dispositivos comunes y aplicaciones. De esta forma cada log generado por un dispositivo en particular puede ser leído y desplegado en otro diferente. También se le considera como aquel mensaje que genera el programador de un sistema operativo, alguna aplicación o algún proceso, en virtud del cual se muestra un evento del sistema (38).

1.5.8.2 Bacula

Es una colección de herramientas de respaldo, capaces de cubrir las necesidades de respaldo de equipos bajo redes IP. Se basa en una arquitectura Cliente-servidor que resulta eficaz y fácil de manejar, dada la amplia gama de funciones y características que brinda; copiar y restaurar ficheros dañados o perdidos. Además, debido a su desarrollo y estructura modular, Bacula se adapta tanto al uso personal como profesional, para parques de ordenadores muy grandes (39).

Estos elementos constituyen además una lista de chequeo que debe utilizar para el diseño y desarrollo de la aplicación (10).

1.6 Soberanía Tecnológica

La soberanía tecnológica es un concepto que se ha introducido en la medida en que los pueblos y estados han ganado en conciencia de la importancia de su seguridad y del impacto de la información y los sistemas en la misma. Se entiende que un estado es soberano tecnológicamente cuando está en completo control de toda su infraestructura tecnológica, en el ámbito de las tecnologías de la información, eso solo se logra a través del software 100% libre de código ilegible (binarios, blobs, privativos, entre otros) (9).

La soberanía tecnológica no se decreta, se construye. No se puede cuestionar o colocar en la palestra de juicios, en virtud de que no está formada o constituida, se construye de acuerdo con las necesidades que surgen en el país y las posibles soluciones que se le acrediten a estas para afianzar el proyecto nacional. La soberanía tecnológica es un concepto que tiene relación con la capacidad que tiene el país para tomar sus decisiones y no depender de aquellas áreas que son estratégicas (10).

La Universidad de Ciencias Informáticas (UCI), no se ha quedado atrás en estos adelantos, primera universidad de código abierto, así llamada desde hace algún tiempo, ya que migrará en el futuro próximo a sistemas operativos de código abierto, convirtiéndose así en el primer centro docente de altos estudios en completar este programa que impulsa el país. Ejemplo de esto es el proyecto NOVA, el sistema operativo cubano basado en GNU/Linux que ya está en su versión 3.0, la que permite un nivel de seguridad informática mucho mayor que los sistemas propietarios, pues son diseñados y programados según las necesidades y características de Cuba.

El producto GESPRO será comercializado bajo licencia GPL, el propio producto y los activos con los que dispondrá son dominados completamente por la UCI, logrando aumentar la seguridad de la información manejada en los proyectos productivos llevados a cabo en la misma y potenciando la independencia tecnológica del modelo de producción del país.

1.7 Conclusiones

De una revisión del estado del arte se puede concluir que, se logró sentar las bases teóricas para un correcto análisis de la arquitectura propuesta. Se profundizó en los conocimientos básicos de la arquitectura de software tales como: los conceptos fundamentales referentes a la misma y las principales corrientes arquitectónicas, sobre las mismas se plasmaron conceptos ofrecidos por los principales conocedores del término. Se realizó un estudio de algunos de los modelos arquitectónicos puestos en práctica en el mundo y se llegó a la conclusión de que el más indicado para el desarrollo del presente trabajo es el propuesto e implantado en la UCI, donde se le confiere especial atención a la Vista de Seguridad, principal objetivo del presente trabajo.

Se abordan además, temas referentes a la vista propuesta, como son los pasos a seguir para su correcta definición en el sistema, tales como: identificación de requisitos

de seguridad del producto,entre otras especificaciones necesarias en la definición de la seguridad propuesta, se hace alusión a las vistas arquitectónicas y cómo se hace necesario separar las vistas de acuerdo a lo que se necesite describir, aspectos de la seguridad informática, así como el papel que juega la soberanía tecnológica en el aseguramiento de la misma.

Capítulo 2: Propuesta Arquitectónica de la Vista de Seguridad.

Introducción

En el presente capítulo se presenta una propuesta de la elaboración de la vista de seguridad del sistema GESPRO 12.05. Para ello se comenzará con la caracterización de la solución vista desde las prácticas para el desarrollo de aplicaciones seguras. Se definirán además, los niveles requeridos en la vista desde la arquitectura propuestos en el Expediente de mejora de la UCI.

2.1 Vistas de Arquitectura de Tecnología

2.1.1 Propuesta de Seguridad

2.1.2 Nivel 1 Aplicación: Identificación de requisitos de seguridad del producto

Para realizar la captura de los requisitos de seguridad, se definieron una serie de preguntas pertinentes para de esta manera lograr una correcta identificación de las características básicas con las que debe contar el sistema, como son: la audiencia que interactúa con la aplicación, por quién se realizará la administración del sistema, así como el entorno en que se desplegará el producto, si brindará algún servicio de infraestructura, entre otras características importantes en la definición del nivel de seguridad con la que contará el producto.

Pregunta 1. ¿Cuál es la audiencia de la aplicación?

Todos los roles definidos en el centro.

Pregunta 2. ¿Qué significa la seguridad para esta audiencia?

La seguridad para esta audiencia es un aspecto fundamental ya que toda la información sensible sobre su institución se encuentra recogida en el sistema informático implantado, cualquier ataque significaría daños irreparables en la empresa, así como la divulgación de información lo cual traería consigo comprometer la integridad, disponibilidad o confidencialidad de la misma.

Pregunta 3. ¿La seguridad es diferente para diferentes miembros de la audiencia?

Sí, en dependencia del rol que desempeñe el usuario va a ser el nivel de acceso a la información de la aplicación.

Pregunta 4. ¿Los requerimientos de seguridad son diferentes para diferentes clientes?

No

Pregunta 5. ¿En qué entorno se desplegará el producto?

- □ Aplicación Web publicada en Internet.
- □ Aplicaciones empotradas en celulares.
- ✓ Aplicación en red cerrada.

Pregunta 6. ¿Qué se está intentando proteger?

Datos sensibles de los proyectos.

Pregunta 7. ¿Qué implicaciones tiene para los usuarios que falle la seguridad?

- Pérdida o cambio de información.
- Divulgación de información sensible.
- Fallo en el sistema.
- Afectación en la calidad del producto.

Pregunta 8. ¿Quién administrará la seguridad del sistema?

El sistema es administrado por la persona que desempeña el rol de administrador, el cual es el encargado de otorgar permisos a los usuarios de la aplicación en dependencia del o los roles que desempeñe en la misma.

Pregunta 9. ¿Qué servicios de infraestructura de seguridad de las que brinda el entorno donde se usará el software pueden ser utilizadas?

- LDAP: Mediante la utilización del protocolo de acceso unificado se protegerá la contraseña de los usuarios que se autentican por medio de este.
- Servicio de notificaciones: La correcta configuración de este servicio permitirá, que solo las direcciones relacionadas con la aplicación GESPRO, establecidas en el servidor de correo de la organización, serán las únicas autorizadas a enviar las notificaciones relacionadas con las peticiones referentes a los proyectos.

2.1.3 Nivel 2 Aplicación: Autenticación y capa de presentación

- ✓ Proveerá algún mecanismo seguro de encriptación y trasferencia de datos.

Para que los datos que viajen por la red no se encuentren en texto plano y no puedan ser vistos por personas que tengan algún programa monitoreando la red en busca de información.

- ✓ Se controlará desde la aplicación de autenticación el acceso a los módulos y subsistemas según el rol que se está autenticando.

Después de la autenticación del usuario este tendrá un rol o roles asociados, los cuales permitirán darle acceso solo a los módulos o partes de la aplicación que tengan afinidad con este rol.

- ✓ Desde los roles que se definirán en la aplicación se garantizará que no sea necesario autenticarse con el rol de administrador para poder tener acceso a las funcionalidades y subsistemas.

No solo el administrador será quien tenga acceso a todas y cada una de las funcionalidades, cada rol podrá acceder a las funcionalidades que sean definidas para ese rol.

- ✓ El sistema de autenticación recogerá datos necesarios de los usuarios y tendrá el control de identidades de los mismos.

El sistema de autenticación se podrá realizar de forma interna o mediante LDAP, en cualquiera de los dos casos la aplicación recogerá los datos del usuario y la contraseña, comprobando que son correctos para validar que tiene permisos para acceder a dicha aplicación.

- ✓ Utilizará alguna de las AAA (Autenticación, Autorización, Auditoría) recomendadas en la vista de la arquitectura de entorno de desarrollo tecnológico.

El sistema utilizará dos A de las tres recomendadas, estas serán autenticación y autorización, la auditoría estará presente pero en posteriores versiones del sistema.

- ✓ Existirán requisitos para la validación correcta de las entrada/salida de datos de cada una de las interfaces de la aplicación.

En los requisitos y casos de pruebas se especifican cuáles son los datos correctos que deben ser brindados a la aplicación, así como los que la misma debe devolver después de realizada una acción.

2.1.4 Nivel 3 Aplicación: Nivel de la capa de negocio

- ✓ En el código se programará el tratamiento de las excepciones.

Las excepciones que puedan ocurrir en la ejecución de la aplicación serán controladas desde el código de la misma, para que estas no afecten el funcionamiento normal del sistema.

- ✓ El tratamiento de las excepciones permitirá un seguimiento hasta guardar información acerca del lugar donde se produjo el error y de los parámetros de configuración del sistema que lo provocaron.

En algunos escenarios de la aplicación el tratamiento de las excepciones o de los errores ocurridos guardará datos sobre el lugar donde se ocasionó, que podrán ser guardados en la aplicación como tal o en logs que se generarán del funcionamiento de la misma, ejemplo de esto se podrá observar en las alertas ocurridas debido a la sincronización entre los proyectos del GESPRO Gerencial y los GESPRO transaccionales creados en un futuro.

El proceso se puede observar a través del siguiente diagrama, representando el tratamiento de excepciones en uno de los escenarios del sistema GESPRO:

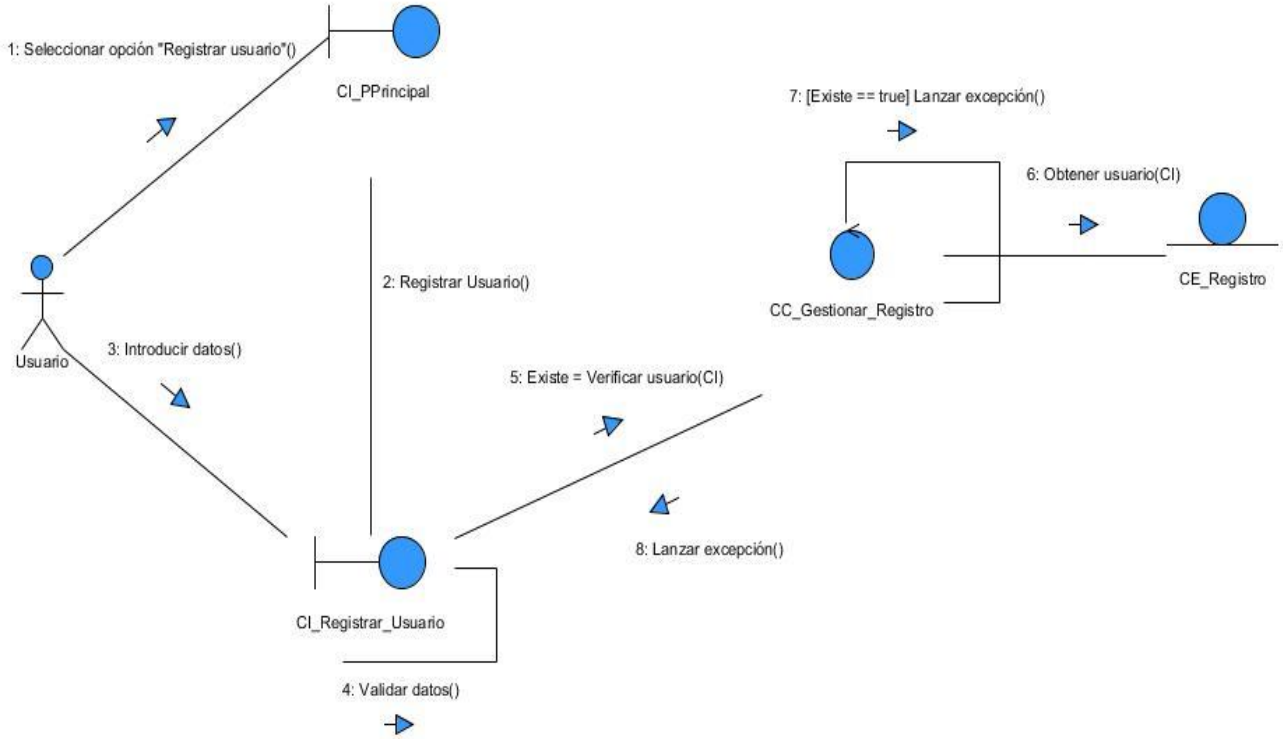


Figura 2. Diagrama de colaboración registrar usuario

La imagen muestra un ejemplo de como será el tratamiento de excepciones en el escenario “Registrar Usuario”, o sea, el usuario selecciona la opción registrar usuario en la página principal, una vez realizada esta acción, el sistema muestra la interfaz “Registrar Usuario”, el mismo valida los datos para verificar que sean correctos, de ser así, verifica si existe en el sistema, obteniendo como resultado que ya fue autenticado previamente, lanzando una excepción especificando dicho error.

- ✓ Se seguirá la traza del trabajo del usuario en la aplicación.

En algunos escenarios se realizará la recopilación de información de la acción del usuario ejemplo de esto se tiene: Peticiones, Cambios, Noticias, Documentos, Archivos, Modificaciones Wiki, Mensajes y Tiempo dedicado.

2.1.5 Nivel 4 Aplicación: Capa de datos

- ✓ Se validarán los datos recibidos de otras aplicaciones externas.

La aplicación podrá recibir datos de otras aplicaciones externas y de otras instancias de la misma aplicación, estos datos serán validados o acotejados según sea el caso, para de esta forma mantener la calidad de los datos que se guardan en ella.

- ✓ Cada usuario del sistema se corresponderá con un usuario de la base de datos.

Los usuarios del sistema siempre se corresponderán con los que estarán guardados en la BD debido a que para poder entrar a la aplicación el usuario deberá realizar un previo registro en la misma para luego entrar con el usuario que especificó.

- ✓ Se restringe el acceso a las bases de datos desde los IP en los cuales correrán los subsistemas que acceden a ellas.

Los servidores de BD contarán con un firewall en este caso *iptables* por tratarse de servidores que están en Ubuntu, en el cual se restringirá el acceso a la misma solo desde el IP de la aplicación que usará dicha BD.

- ✓ La aplicación se conectará a la base de datos utilizando usuarios diferentes o a través de un único usuario para la conexión.

Sin importar el usuario que esté interactuando con la aplicación, esta se conectará a la BD con un único usuario definido en la configuración de la misma.

- ✓ Se controlarán los niveles de seguridad a nivel del usuario de la base de datos.

Al usuario de la BD le será asignado un rol, que tendrá privilegios en dependencia de las acciones que puede realizar en la aplicación.

2.1.6 Nivel 5 Aplicación: Seguridad del Servidor de aplicación

En esta sección se muestran todos los niveles en los que se puede desempeñar una aplicación y la especificación del estado de la seguridad de la aplicación vista desde el entorno:

- □ Los servidores donde correrá la aplicación están accesibles directamente desde internet.
- □ Existirá un firewall FW-1 que limitará el acceso externo desde internet a la red de la organización.

- Existirá una zona desmilitarizada DMZ detrás del firewall FW-1 y donde la organización ubicará los servicios y software comunitarios para el acceso desde internet.

- Existirá un firewall FW-2 en la DMZ donde se controlará el acceso a la red interna y a los servidores internos de la aplicación.

✓ Existirán un grupo de servidores detrás del firewall FW-2 que controlarán el acceso a los servidores de la organización.

En la universidad existe un nivel por encima del proyecto que es el encargado del acceso desde internet de agentes externos a la red del mismo. Es por esto que lo que se utilizará será un grupo de servidores detrás del firewall FW-2 que controlará el acceso a los servidores del proyecto.

2.1.7 Nivel 6 Aplicación: Tecnología para la ofuscación de código

El sistema utilizará para la protección del código vistas desde la ofuscación, la tecnología:

◦ ODEC (CD HLG), para soluciones en PHP.

◦ *Visual Studio Ofuscator*, para soluciones en .NET.

✓ No aplica.

◦ Ninguna de las anteriores.

La ofuscación de código dinámica es una técnica que transforma el código en garabatos incomprensibles, esta técnica puede ser utilizada con dos propósitos principales, el primero como una herramienta de protección de derechos de autor del código de los programas, y el segundo como una amenaza a los mecanismos actuales de seguridad en la red.

La ofuscación del código dificulta la posibilidad de obtener el código original y por tanto la realización de ingeniería inversa.

Al ser GESPRO una herramienta de código libre, no debe contar con esta característica, ya que entre los objetivos de la misma no se encuentra la privatización, ni de la herramienta como tal, ni del código implementado por su equipo de desarrollo.

2.1.8 Nivel 7 Aplicación: Tecnología para el Sellado de Código (compilación)

El sellado de código, o conversión a código cerrado, transforma el componente de software en código nativo o algún *byte-code*.

El sistema pudiera utilizar para la protección del código vistas desde el sellado del código, la tecnología:

- *HipHopfor* PHP, para soluciones en PHP
- J++, para soluciones en Java
- ✓ No aplica
- Ninguna de las anteriores

El Sellado de Código se caracteriza por ser muy similar a la Ofuscación de Código, o sea, que posee la misma finalidad, ocultar el código implementado por el programador, con el objetivo de evitar la copia o futuro uso del mismo, por tanto tampoco es aplicable a la herramienta propuesta.

2.1.9 Nivel 8 Aplicación: Monitoreo de la seguridad y proceso de desarrollo seguro

- ✓ Se guardarán las trazas en el funcionamiento de la aplicación en forma de logs.

Logs. (Archivo donde se guardará el funcionamiento de la aplicación).

- ✓ Se establecerá como estrategia para la rotación de los logs de las trazas de la aplicación, mantener una copia de los mismos por **10 días.**

No cumple objetivo además que por el espacio que esto ocupará, que los logs de la aplicación se guarden por más de 10 días, por lo que estos se rotarán usando el *logrotate* del sistema operativo.

- ✓ Se guardarán las trazas en el funcionamiento de la base de datos en forma de logs.

El funcionamiento de la BD se guardará en trazas en forma de log, debido a que la BD a utilizar será PostgreSQL y la instalación de la misma trae esto por defecto.

- ✓ Se establecerá como estrategia para la rotación de los logs de las trazas de la BD, mantener una copia de los mismos por **10 días.**

No cumple objetivo además de por el espacio que esto ocupará, que los logs de la BD se guarden por más de 10 días, por lo que estos se rotarán usando el *logrotate* del sistema operativo.

- ✓ Se realizarán copias de seguridad de las bases de datos, del repositorio, así como de cualquier otra información vital.

Se realizarán salvas de la BD de la aplicación junto con otros archivos relacionados con la misma para tener respaldos en caso de catástrofe, estas salvas se guardarán por el periodo de un mes.

- ✓ Se utilizará el Bacula para realizar las copias de seguridad de los repositorios.

Se seleccionó esta herramienta por ser libre y estar incluida en los repositorios de Ubuntu.

- ✓ Compromiso de la alta gerencia: existirá en el entorno un plan de seguridad informática que recogerá este documento de arquitectura de las aplicaciones como parte de la seguridad

- ✓ Se identificarán los riesgos para la seguridad en el plan de gestión de riesgos del proyecto.

2.2 Conclusiones

Durante la realización de este capítulo se logró desarrollar la propuesta de solución a la Vista de Seguridad de la Arquitectura del Sistema GESPRO 12.05. Para ello se comenzó identificando los requisitos de seguridad del producto. Se definieron además, las funcionalidades con las que debe cumplir el sistema para lograr que este sea un sistema seguro, así como el monitoreo de la aplicación para lograr un proceso de desarrollo con la seguridad requerida.

Capítulo 3: Validación práctica de la propuesta realizada.

Introducción

En el presente capítulo se culmina la propuesta de solución de la elaboración de la Vista de Seguridad de GESPRO 12.05. Para ello se demostrará la calidad del producto a través de una comparación de la misma con las anteriores utilizadas en la UCI. Se validará la propuesta demostrándose la factibilidad de la misma con la obtención de los resultados que prueban la necesidad de su puesta en práctica en la universidad. Por cada nivel se identificaron variables para establecer la comparación entre las herramientas y finalizar con la validación de la propuesta. Además se validará la propuesta poniendo en práctica el método de evaluación preexperimento, para arribar a resultados del nivel de factibilidad existente en la aplicación de la misma.

3.1 Desglose de las variables de comparación.

Para realizar una comparación entre las herramientas de gestión de proyectos informáticos existentes en la UCI antes de implantado el sistema GESPRO, se analizaron solo tres herramientas, escogidas por su mayor nivel de utilización por los proyectos productivos de la universidad, en el caso particular del Redmine, no se tuvo en cuenta por estar GESPRO basado en esta herramienta.

Una entrevista realizada a especialistas de CALISOFT (30) mostró como resultado, que del total de proyectos de la universidad, un 2 % empleaba la herramienta *Team Foundation Server*, el 21 % *GForce*, mientras que un 16 % *Redmine* y otro 15 % el *Trac*. Para *Bugzilla*, *DotProject*, *Jira* y *Microsoft Team System* se obtenía un 1 % de uso respectivamente. Se reveló además que el 42 % restante de los proyectos no utilizaban ninguna herramienta.

Para realizar una comparación entre las herramientas existentes en la universidad y la propuesta en el presente trabajo, se identificaron variables por cada nivel, especificando el cumplimiento de las herramientas analizadas con las mismas:

Nivel 1 Aplicación: Identificación de requisitos de seguridad del producto

- Utilización de servicios de infraestructura
  - ✓ GESPRO
  - ✓ Team Foundation Server
- Requisitos de seguridad.
  - ✓ GForce
  - ✓ GESPRO
  - ✓ Team Foundation Server
  - ✓ Trac

Nivel 2 Aplicación: Autenticación y capa de presentación

- Encriptación y trasferencia de datos
  - ✓ GForce
  - ✓ GESPRO
  - ✓ Team Foundation Server
  - ✓ Trac
- Protocolo seguro de acceso
  - ✓ GESPRO
  - ✓ Team Foundation Server
- Marcas de tiempo
- Auditoría de trazas
  - ✓ Team Foundation Server
- AAA
  - ✓ GESPRO (El caso particular de las AAA GESPRO solamente utiliza dos de ellas, ya que no tiene en cuenta la auditoría, por tanto en este punto resto un 5%)
  - ✓ Team Foundation Server
  - ✓ Trac (El caso particular de las AAA Trac solo utiliza dos de ellas, la misma no tiene en cuenta la auditoría, por tanto en este punto se resta un 5%)
- Mapa de navegación

- ✓ Team Foundation Server

Nivel 3 Aplicación: Nivel de la capa de negocio

- Tratamiento de excepciones
  - ✓ GForce
  - ✓ GESPRO
  - ✓ Team Foundation Server
  - ✓ Trac

Nivel 4 Aplicación: Capa de datos

- Validación de los datos recibidos de otras aplicaciones externas
  - ✓ GForce
  - ✓ GESPRO
  - ✓ Team Foundation Server

Nivel 5 Aplicación: Seguridad del Servidor de aplicación

- Seguridad del Servidor de aplicación
  - ✓ GForce
  - ✓ GESPRO
  - ✓ Team Foundation Server
  - ✓ Trac

Nivel 6 Aplicación: Tecnología para la ofuscación de código

- Ofuscación de código
  - ✓ Team Foundation Server

Nivel 7 Aplicación: Tecnología para el Sellado de Código (compilación)

- Sellado de Código
  - ✓ Team Foundation Server

Nivel 8 Aplicación: Monitoreo de la seguridad y proceso de desarrollo seguro

- Se guardan las trazas

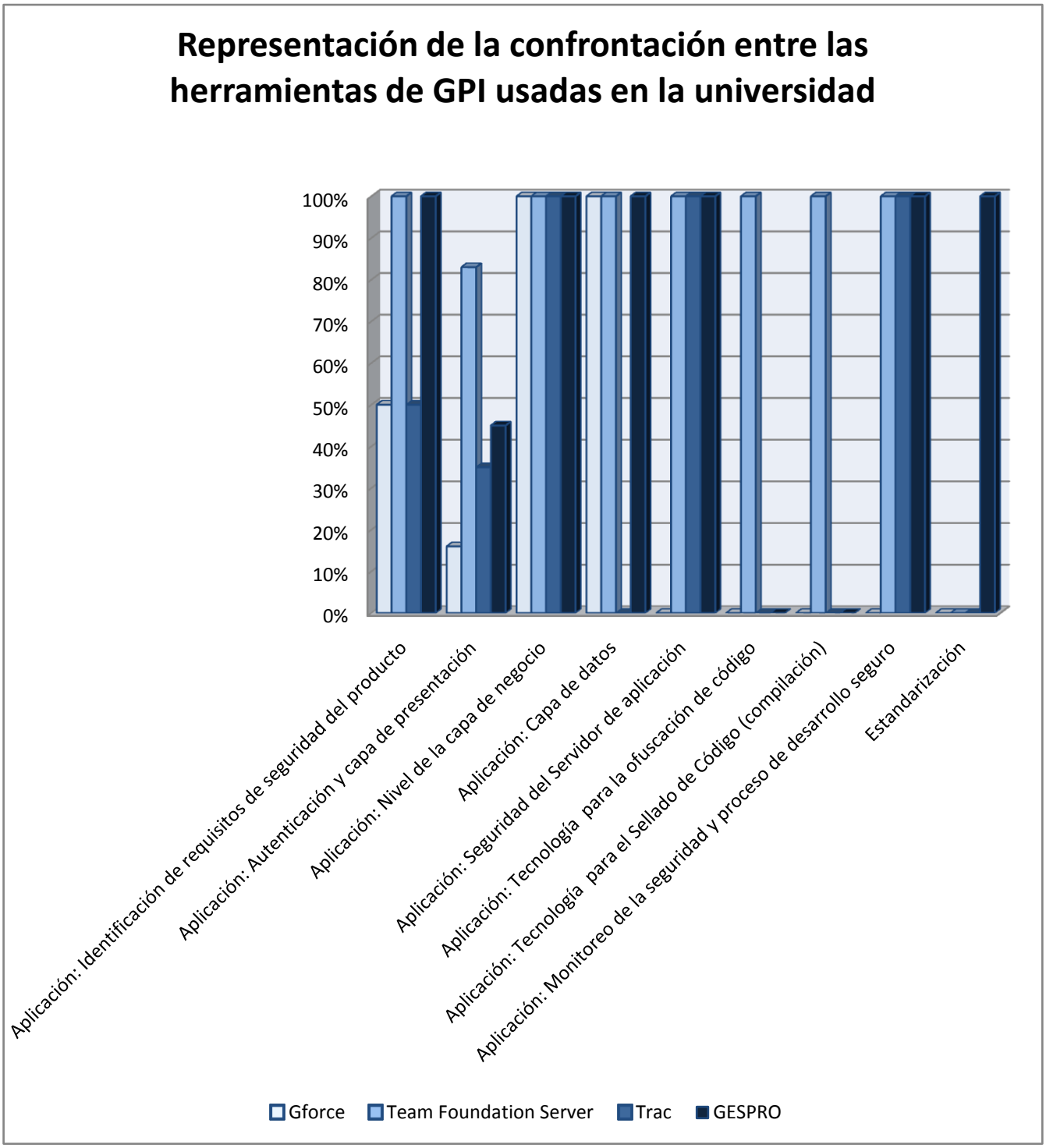
- ✓ GESPRO
- ✓ Team Foundation Server
- ✓ Trac

3.2 Confrontación entre las herramientas de GPI usadas en la universidad.

Como resultado del desglose de las variables de comparación se realizó un cálculo de por ciento de cumplimiento de las herramientas analizadas por cada elemento contenido en el nivel, obteniéndose como resultado la siguiente tabla:

	GForce	Team Foundation Server	Trac	GESPRO
Nivel 1 Aplicación: Identificación de requisitos de seguridad del producto	50%	100%	50%	100%
Nivel 2 Aplicación: Autenticación y capa de presentación	16%	83%	35%	45%
Nivel 3 Aplicación: Nivel de la capa de negocio	100%	100%	100%	100%
Nivel 4 Aplicación: Capa de datos	100%	100%	0%	100%
Nivel 5 Aplicación: Seguridad del Servidor de aplicación	0%	100%	100%	100%
Nivel 6 Aplicación: Tecnología para la ofuscación de código	0%	100%	0%	0%
Nivel 7 Aplicación: Tecnología para el Sellado de Código (compilación)	0%	100%	0%	0%
Nivel 8 Aplicación: Monitoreo de la seguridad y proceso de desarrollo seguro	0%	100%	100%	100%
Estandarización	0%	0%	0%	100%

Tabla 1. Confrontación entre las herramientas de GPI usadas en la universidad.



A través de la gráfica se puede observar como el por ciento de cumplimiento con los requerimientos identificados en cada nivel, son superados por GESPRO, demostrando el alto por ciento de seguridad que posee el mismo, mostrando así la factibilidad del uso de esta herramienta en la universidad.

3.3 Validación práctica de la implantación.

Para realizar la validación práctica de la implantación de la propuesta, se analizaron por separado las dos variables dependientes. Como resultado de la confrontación de las herramientas de gestión de software, se obtuvo como resultado para la seguridad, el siguiente por ciento de factibilidad del uso de las herramientas analizadas en la universidad.

3.3.1 Seguridad.

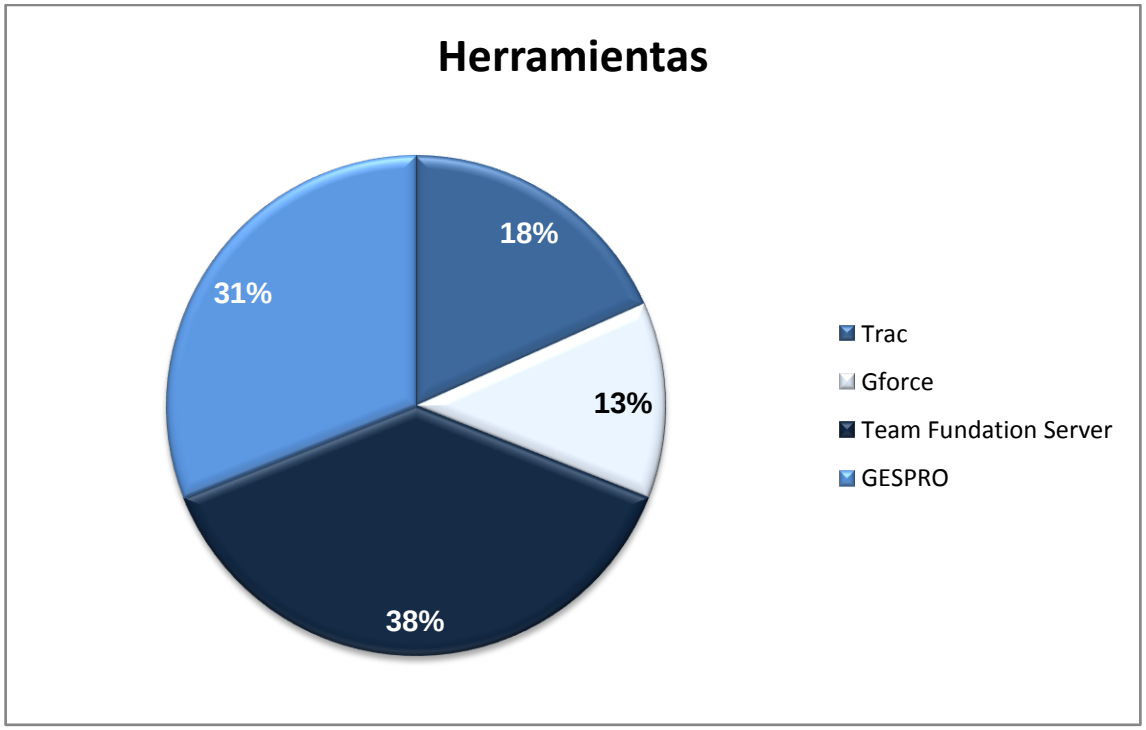


Figura 3. Por ciento de factibilidad de uso de la herramienta respecto a la seguridad

El resultado obtenido muestra claramente el alto nivel de factibilidad del uso de la herramienta para gestión de proyectos GESPRO. Aunque la herramienta *Team Fundation Server* muestra mayor por ciento que la propuesta, no se tiene en cuenta para la implantación como herramienta única en la universidad, por ser de licencia privada y no cumplir con uno de los principales objetivos del centro, que es lograr la soberanía tecnológica, evitando así, la dependencia de terceros países para fabricar sus sistemas informáticos.

3.3.2 Soberanía Tecnológica.

Para realizar la validación práctica de la soberanía tecnológica lograda con la implantación de la herramienta GESPRO para la gestión de proyectos informáticos, se analizaron los tipos de licencias de las herramientas, así como el riesgo de convertirse en herramientas propietarias en caso de ser software libre, y la capacidad del completo dominio de la misma en la organización.

	Licencia Privativa	Riesgo de que quién lo desarrolla pueda privatizarlo en caso de que sea software libre	Capacidad en su organización si usa este producto de dominarlo completamente
GForce	No	Medio	No
Team Foundation Server	Si	---	No
Trac	No	Bajo	No
GESPRO	No	Muy Bajo	Si

Tabla 2. Comparación de las licencias de los productos sustitutos del desarrollado y sus propietarios

Se obtuvo como resultado el alto nivel de cumplimiento del sistema GESPRO de acuerdo con los aspectos más importantes a destacar en aras de lograr una completa soberanía tecnológica de la herramienta para la gestión de proyectos, propuesta a implantar en la universidad.

El proyecto además persigue como objetivo, sustituir los productos actualmente existentes en la universidad, por tanto es parte de la estrategia, identificar todos estos elementos que son fortalezas del mismo.

Para ello se realizó un estudio de los escenarios en los que se utiliza la aplicación.

	MININT	CENTRO FORTES UCI	CENTRO CDAE UCI
Hay seguridad de que el sistema no tiene puertas traseras.	Sí	Sí	Sí
Sistema supervisable. El sistema debe archivar en la base de datos todos o parte de los datos registrados en el historial.	Sí	Sí	Sí
¿Se sigue la traza del trabajo del usuario en la aplicación?	Sí, de algunas acciones	Sí, de algunas acciones	Sí, de algunas acciones
Se guardan las trazas en el funcionamiento de la aplicación en forma de logs y el sistema permite su análisis.	Sí, No	Sí, No	Sí, No
Se guardan las trazas en el funcionamiento de la base de datos en forma de logs y el sistema permite su análisis.	Sí, No	Sí, No	Sí, No
Se realizan copias de seguridad de las bases de datos, del repositorio, así como de otra información vital.	Sí	Sí	Sí

¿Se validan los datos recibidos de otras aplicaciones externas?	Sí	Sí	Sí

Tabla 3. Estudio de los escenarios en los que se utiliza la aplicación.

3.4 Oportunidad de mejora.

3.4.1 Nivel 2 Aplicación: Autenticación y capa de presentación

- ✓ La herramienta de autenticación no contiene marcas de tiempo asociadas al proceso de autenticación, lo cual no se ha desarrollado aún, sin embargo se considera que debe implementarse porque aumentaría el nivel de seguridad del sistema, debido a que evita el acceso de personal no autorizado a información sensible.

- ✓ La herramienta de autenticación tiene marcas de tiempo asociadas al proceso de autenticación.

- ✓ La herramienta de autenticación no permite la auditoría de trazas de las acciones de autenticación por usuarios, dando lugar a la no existencia de un registro de las acciones del usuario, limitando en este sentido la seguridad del sistema por el desconocimiento de las acciones realizadas en el mismo.

3.4.2 Nivel 4 Aplicación: Capa de datos

- ✓ Existen configuraciones a nivel de servidor de bases de datos que permite controlar la inyección de código.

La inyección SQL consiste en la modificación del comportamiento de las consultas mediante la introducción de parámetros no deseados en los campos a los que tiene acceso el usuario. Este tipo de errores puede permitir a usuarios malintencionados acceder a datos a los que de otro modo no tendrían acceso y, en el peor de los casos, modificar el comportamiento de las aplicaciones (14).

En el sistema no existen configuraciones a nivel de servidor de bases de datos que permitan controlar la inyección de código, lo cual podría provocar baches en la

seguridad. Con estas inyecciones se pueden obtener datos escondidos, eliminar o modificar tablas de la base de datos.

3.4.3 Nivel 8 Aplicación: Monitoreo de la seguridad y proceso de desarrollo seguro

- ✓ Realizar auditorías al código del servidor para la detección de cambios en el código.
- ✓ Utilizar el Sistema Análisis Estático de Vulnerabilidades (SAEV) (Centro TLM).
- ✓ Utilizar la herramienta para las Auditorías de Bases de Datos y Sistemas Operativos (AuditBD) (Centro TLM).
- ✓ Comprobar el entorno con el sistema de escaneo de Vulnerabilidades Online v2 (Centro TLM).

3.5 Conclusiones

Con la realización de este capítulo se finalizó la propuesta de solución de la Vista de Seguridad de la Arquitectura del Sistema GESPRO 12.05. Para ello se comprobó la calidad del producto a través de una comparación de la misma con las anteriores utilizadas en la UCI. Se logró además validar la propuesta demostrándose la factibilidad de la misma con la obtención de los resultados que prueban la necesidad de su puesta en práctica en la universidad.

Conclusiones generales

La investigación desplegada en la realización de este trabajo de diploma y el análisis de la documentación estudiada permitió el cumplimiento de los objetivos planteados en los inicios del mismo.

Se desarrolló la propuesta de solución para la Vista de Seguridad de la Arquitectura del Sistema GESPRO, caracterizándose la solución vista desde los niveles definidos por el Expediente de mejora de la UCI.

A través de la implantación de la propuesta fue posible demostrar la factibilidad de la misma, ya que se logró aumentar la seguridad de la información sensible manejada por la universidad, así como contribuir en el desarrollo de la soberanía tecnológica del país.

La investigación realizada demostró el alto por ciento de factibilidad del uso de la herramienta GESPRO en la universidad teniendo en cuenta el cumplimiento de los niveles de seguridad, lo cual fue validado mediante el método del preexperimento.

Recomendaciones:

- La vista está muy enfocada a alguna herramienta como tal, se recomienda estandarizarla más.
- Se recomienda que el resultado propuesto en la investigación sea implantado en todos los Sistemas de Gestión de la universidad.
- Se recomienda añadir al sistema la realización de auditorías propuestas por las AAA.
- Analizar el artefacto generado, de tal manera que su uso favorezca a la línea de vida del proyecto y facilite el trabajo de los desarrolladores.
- Actualizar cada cambio que surja a medida que el desarrollo del proyecto avance para lograr una descripción que respalde todo el desarrollo del software.

Glosario de términos

ADLs: (*Architecture Description Language*) Lenguajes de Descripción Arquitectónica, lenguaje descriptivo de modelado que se focaliza en la estructura de alto nivel de la aplicación antes que en los detalles de implementación de sus módulos concretos.

ARPANET: (*Advanced Research Projects Agency Network*) red de computadoras.

ASN.1: *Abstract Syntax Notation One* (notación sintáctica abstracta 1, ASN.1) es una norma para representar datos independientemente de la máquina que se esté usando y sus formas de representación internas. Es un protocolo de nivel de presentación en el modelo OSI.

BD: Base de Datos.

Bugzilla: Es una herramienta basada en Web de seguimiento de errores (Bug Tracking System o BTS).

Byte-code: Es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina.

CMM: Modelo de evaluación de los procesos de una organización. La visión general de los modelos basados en la madurez de las capacidades: Modelo de Capacidad y Madurez.

DMZ: (Zona Desmilitarizada) Los sistemas Firewall permiten definir las reglas de acceso entre dos redes. Sin embargo, en la práctica, las compañías cuentan generalmente con varias subredes con diferentes políticas de seguridad. Por esta razón, es necesario configurar arquitecturas de firewall que aíslen las diferentes redes de una compañía. Esto se denomina "aislamiento de la red".

DotProject: Es una completa herramienta que permite gestionar las distinta fases y tareas que componen un proyecto.

FTP: (*File Transfer Protocol*) Permite transferir archivos locales hacia un servidor web.

GForce: Es un software libre basado en la Web para la gestión colaborativa de proyectos de software.

GPL: La Licencia Pública General de GNU o más conocida por su nombre en inglés *GNU General Public License* o simplemente sus siglas del inglés GNU GPL, está orientada principalmente a proteger la libre distribución.

GPI: Gestión de Proyectos Informáticos.

HTTP: *Hypertext Transfer Protocol* o HTTP (protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la *World Wide Web*.

IEEE: Es un organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional.

Jira: Es una aplicación basada en web para el seguimiento de errores, de incidentes y para la gestión operativa de proyectos. Jira también se utiliza en áreas no técnicas para la administración de tareas.

LAN: Una red de área local, red local o LAN (*local areanetwork*) es la interconexión de una o varias computadoras y periféricos. Su extensión está limitada físicamente a un edificio o a un entorno de 200 metros, con repetidores podría llegar a la distancia de un campo de 1 kilómetro.

LDAP: (*Lightweight Directory Access Protocol*) Protocolo Ligero de Acceso a Directorios que hacen referencia a un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

Log: Registro.

Logrotate: Es un registro de eventos producidos durante un periodo de tiempo concreto.

OSI: (*Open System Interconnection*) modelo de interconexión de sistemas abiertos, es el modelo de red descriptivo creado por la Organización Internacional para la Estandarización.

RUP: Es una de las principales compañías que lleva la delantera en lo referente a Arquitectura de Software.

RC4: Dentro de la criptografía RC4 o ARC4 es el sistema de cifrado de flujo *Streamcipher* más utilizado y se usa en algunos de los protocolos más populares como *Transport Layer Security* (TLS/SSL) (para proteger el tráfico de Internet) y *Wired Equivalent Privacy* (WEP) (para añadir seguridad en las redes inalámbricas).

RSA: Es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

Redmine: Es una herramienta para la gestión de proyectos que incluye un sistema de seguimiento de incidentes con seguimiento de errores.

SEI: (*Software Engineering Institute*) Instituto Federal Estadounidense de investigación y desarrollo, fundado por el Congreso de los Estados Unidos en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software, que dieran respuesta a los problemas que generaba al ejército estadounidense la programación e integración de los sub-sistemas de software en la construcción de complejos sistemas militares. Financiado por el Departamento de Defensa de los Estados Unidos y administrado por la Universidad Carnegie Mellon.

SMTP: (*Simple Mail Transfer Protocol*) Protocolo Simple de Transferencia de Correo, es un protocolo de la capa de aplicación. Protocolo de red basado en textos utilizados para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, entre otros).

SQL: El lenguaje de consulta estructurado (*structured query language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas.

Team Foundation Server: Ofrece funciones de control de código fuente, seguimiento de elementos de trabajo.

Trac: Es una herramienta para la gestión de proyectos y el seguimiento de errores escrita en Python, inspirado en CVSTrac.

UML: (*Unified Modeling Language*) Lenguaje Unificado de Modelado.

Bibliografía

1. **Arquitectura de Software.** Arquitectura de Software. [En línea] Diciembre de 2006. [Citado el: 20 de 1 de 2012.] http://arquitectura-de-software.blogspot.com/2006/12/qu-es-un-arquitecto-de-software.html..

2. **Centro de e-Learning.** Centro de e-Learning. [En línea] [Citado el: 15 de Enero de 2012.] http://www.sceu.frba.utn.edu.ar/e-learning/cursos-a-distancia/Inform%C3%A1tica/Arquitecturas-de-Software/temario.

3. **Conceptos básicos sobre Arquitectura de Software.** Grupo de Trabajo Académico GNU/Linux - GLUD. [En línea] 12 de Septiembre de 2011. [Citado el: 10 de Diciembre de 2011.] http://www.slideshare.net/glud/conceptos-basicos-arquitectura-de-software.

4. **Arlet Carrascoso, Yoan.** Gestiopolis. [En línea] 2009. [Citado el: 20 de Febrero de 2012.] http://www.gestiopolis.com/administracion-estrategia/erp-arquitectura-orientada-a-servicios.htm..

5. **Reynoso, Carlos Billy.***Introducción a la Arquitectura de Software.* Buenos Aires : s.n., 2004.

6. **Faneyth Martínez, Luis Alejandro.** Huntingbears. [En línea] 2011. [Citado el: 24 de Marzo de 2012.] http://www.huntingbears.com.ve/el-camino-a-la-soberania-tecnologica-en-venezuela.html.

7. **Lage, Agustín.** Reflexiones sobre el Desarrollo de la Biotecnología en Europa y América Latina. [En línea] Centro de Inmunología Molecular. [Citado el: 19 de Enero de 2012.] http://www.cuba.cu/ciencia/acc/anales9.htm. .

8. **Trucos Windows.** Trucos Windows. [En línea] 8 de Agosto de 2004. [Citado el: 25 de Enero de 2012.] http://www.trucoswindows.net/conteni5id-48-SEGURIDAD-Resumen-de-los-tipos-de-ataques-mas-comunes.html.

9. **Jorrín, MSc. Michaell González, y otros.***Modelo para diseñar arquitecturas de software durante el desarrollo de sistemas de información.* 2011.

10. **Ministerio del Poder Popular para Ciencia, Tecnología e Innovacón.** Ministerio del Poder Popular para Ciencia, Tecnología e Innovacón. [En línea] 7 de Enero de 2009. [Citado el: 8 de Abril de 2012.] http://www.mct.gob.ve/Noticias/2852.

11. **Bases de Datos.** Bases de Datos. [En línea] [Citado el: 16 de Febrero de 2012.] http://www.maestrosdelweb.com/editorial/inyecsql.

12. **Díaz Verdecia, Osvaldo, Quevedo, Campins y Virgen, Damaris.** 2009.

13. **Correa, Reinier Asencio y Hinojosa Calzada, Jeanders Silvio.***Definición de la arquitectura del sistema alas ARBOGEN 2.0.* 2010.

14*GESPRO 11.05 UN SISTEMA PARA LA DIRECCIÓN INTEGRADA DE PROYECTOS PARA LA GESTIÓN DE LA PRODUCCI*2011

15. **Pérez, MsC. Antonio Rodríguez y León Burguera, Lic. Olga Lidia.** GstioPolis.com. [En línea] 14 de 07 de 2009. [Citado el: 6 de Mayo de 2012.] http://www.gestiopolis.com/administracion-estrategia/seguridad-informatica-y-su-control.htm.

16. —. GestiPolis.com. [En línea] 14 de 07 de 2009. [Citado el: 7 de Mayo de 2012.] http://www.gestiopolis.com/administracion-estrategia/seguridad-informatica-y-su-control.htm.

17. **Reynoso, Carlos Billy.***Introducción a la Arquitectura de Software.* BUENOS AIRES : s.n., 2004.

18. **Ochoa, Ing. René Lazo.***Modelo de referencia para el desarrollo arquitectónico de sistemas de software en dominios de gestión.* 2011.

19. —. *Modelo de referencia para el desarrollo arquitectónico de sistemas de software en dominios de gestión.* 2011.

20. **Blanco, Carlos y Ruiz, Francisco.** INGENIERÍA DEL SOFTWARE I. [En línea] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/materiales-de-clase-1/is1_t04_requisitos.pdf.

21. **Vela, José Emiro.** Introducción a Redes. [En línea] 2011. http://www.monografias.com/trabajos/introredes/introredes.shtml.

22. University of Cambridge. [En línea] 20 de Diciembre de 2001. http://www.webcitation.org/5tP0nKIlL.

23. SPL Sistemas de Información, SL. [En línea] 2010. http://www.spl-ssi.com/?sec=articulos&subsec=descripcion&v=aplicaciones_web.

24. **Terrasa, Silvia, Balbastre, Patricia y Crespo, Alfons.** Experiencia docente en el desarrollo de aplicaciones empotradas. [En línea] 2009. http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2002/Cac539_543.pdf.

25. **Pradas, Rafael Calzada.** Introducción al Servicio de Directorio. [En línea] http://www.rediris.es/ldap/doc/ldap-intro.pdf.

26. GuíaWeb 1.0. [En línea] 2011. http://www.guiaweb.gob.cl/guia/glosario.htm#Enc001.

27. Uso de seguridad AAA para la administración de Conversores de Medios. [En línea] 2012. http://www.perlesystems.es/supportfiles/AAA-Security.shtml.

28. Introducción al Manejo de Excepciones en C. [En línea] 2011. http://www.coders.me/c/introduccion-al-manejo-de-excepciones-en-c.

29. **Camejo, René R. Bauta y Galvez, Ariel Torres.***HERRAMIENTA PARA EL REGISTRO Y MONITOREO DE TRAZAS EN EL SISTEMA DE GESTIÓN INTEGRAR CEDRUX.* La Habana : s.n., 2010.

30. **García, J. Lugo.***Herramientas para la Gestión de Proyectos en la UCI.* 2011.

31. **Marañón, Gonzalo Álvarez.** Web Seguro. [En línea] 2010. http://www.iec.csic.es/criptonomicon/ssl.html.

32. Maestros en la Web (Inyección de Código). [En línea] http://www.maestrosdelweb.com/editorial/inyecsql/.

33. **INGHAM, KENNETH y FORREST, STEPHANIE.***A History and Survey of Network Firewalls.* 2009.

34. Buenas Tareas. [En línea] 2010. http://www.buenastareas.com/ensayos/Ofuscacion-De-Codigo/2896717.html.

35. **2000, IEEE Std 1471-.***Recommended Practice for Architectural Description for Software-Intensive Systems.*

36. Ingeniería Técnica en Informática de Sistemas. [En línea] http://gva1.dec.usc.es/~antonio/docencia/2005bd/teoria/PDSeguridad_gris.pdf.

37. **Buschmann, Frank, y otros.***Pattern-oriented software. A system of patterns. s.l. : John Wiley & Sons.* 1996.

38. Apache. Http SERVER PROYECT. [En línea] 2012. http://httpd.apache.org/docs/.

39. Bacula-Web. [En línea] 2012. http://www.bacula-web.org/.

40. **Pérez, MsC. Antonio Rodríguez y Lic. Olga Lidia León Burguera.** GestioPoli.com. [En línea] 14 de 07 de 2009. http: //www.gestiopolis.com/administracion-estrategia/seguridad-informatica-y-su-control.htm.