ANÁLISIS COMPARATIVO ENTRE METODOLOGÍAS PARA EL DESARROLLO SOFTWARE SEGURO DE ACUERDO CON EL ESTÁNDAR ISO/IEC 15408

NANCY SORAIDA BAYONA GUIO

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
DUITAMA
2020

ANÁLISIS COMPARATIVO ENTRE METODOLOGÍAS PARA EL DESARROLLO SOFTWARE SEGURO DE ACUERDO CON EL ESTÁNDAR ISO/IEC 15408

NANCY SORAIDA BAYONA GUIO

Monografía presentada para optar por el título de
ESPECIALISTA EN SEGURIDAD INFORMÁTICA

Msc. ALEJANDRO CARRILLO

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA - ECBTI
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
DUITAMA
2020

NOTA DE ACEPTACIÓN

Firma del Presidente de Jurado

Firma del Jurado

Firma del Jurado

Duitama., 01 de octubre de 2020

DEDICATORIA

Dedico este trabajo primero que todo a Dios, quien inspiro mi espíritu para concluir con esta monografía. A mis padres, que con su apoyo me acompañaron en cada etapa de mi vida han sido mi fortaleza para seguir adelante con mis objetivos profesionales.

AGRADECIMIENTOS

El siguiente trabajo agradezco a Dios por ser mi guía quien me acompañado en este camino, dándome la fortaleza para continuar con mis metas trazadas y a mis padres que han sido mi apoyo incondicional en este reto.

De manera muy especial a mi director del proyecto MSc. Alejandro Carrillo, por haberme guiado durante la elaboración de esta monografía gracias a su amplia experiencia y conocimientos me orientaron al desarrollo y culminación de este te trabajo.

También extiendo un agradecimiento a la Universidad Nacional Abierta y a Distancia UNAD, por darme la oportunidad de enriquecer mi conocimiento profesional.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

DESARROLLO SEGURO se define como una necesidad al momento del diseño y codificación de una aplicación como base fundamental la seguridad en el momento en que se inicia el ciclo de vida del software.

SEGURIDAD DE SOFTWARE está diseñada para proteger el software ante cualquier ataque ya sea por cualquier malware.

SEGURIDAD INFORMÁTICA Se define como una rama dentro de la tecnología de la información que tiene como, función principal la protección de los activos a través de una red.

ESCALA COCKBURN es un método para describir en cuanto procesos se requiere para iniciar un proyecto de software, este método se puede aplicar para cualquier marco de trabajo.

PYME hace referencia a una empresa pequeña o mediana con base al número de ingresos o por número de colaboradores.

INYECCIÓN SQL se define como un método infiltración al código fuente o a base de datos, donde se realice algún tipo de afectación ya sea a nivel de dato o de logia de negocio.

CÓDIGO FUENTE conjunto de líneas de texto que se encuentran de una manera secuencial, con el fin de ejecutar un proceso dentro de una aplicación.

CICLO DE VIDA DE DESARROLLO se define como una estructura aplicada dentro del desarrollo donde se deben seguir una serie de fases que se requiere para terminar un producto con estándares de calidad.

REQUISITO es un es una especificación completa de un comportamiento de un proceso o sistema que se va a desarrollar.

RESUMEN

Investigaciones ha demostrado en los últimos tiempos que la gran mayoría de las vulnerabilidades dentro de los sistemas de información se ejecuta a través del código, debido que se presenta una debilidad dentro de los equipos de ingeniería, donde no se aplica buenas prácticas de desarrollo seguro, dado que para muchos desarrolladores existe un desconocimiento sobre, la existencia de metodologías para el desarrollo de software seguro, de ahí surge la necesidad de indagar que metodologías de desarrollo seguro existe, como se podrían implementar y aplicar dentro de los equipos de desarrollo de acuerdo al número de integrantes; por consiguiente, se realiza un análisis comparativo, entre siete metodologías de desarrollo de software seguro, bajo un estándar internacional Common Criteria (ISO/IEC 15408). A lo largo de este estudio se desarrolla tres capítulos, que permiten describir conceptos, antecedentes, características y finalmente realizar un análisis comparativo entre las metodologías de desarrollo de software seguro y como objeto de estudio, se observa que la metodología de desarrollo seguro, que cumple con la mayoría de los criterios establecidos, dentro del estándar internacional es la metodología Security Development Lifecycle SDL. La investigación permite concluir que hay criterios que hasta el momento ninguna de las metodologías los tiene presentes dentro sus fases, por consiguiente, se considera que estos aspectos se puedan tener en cuenta para futuras investigaciones.

ABSTRACT

Research has shown in recent times that the vast majority of vulnerabilities within information systems are executed through code, due to a weakness within engineering teams, where good safe development practices are not applied, Given that for many developers there is a lack of knowledge about the existence of methodologies for the development of secure software, hence the need to investigate what secure development methodologies exist, how they could be implemented and applied within development teams according to the number of participants; therefore, a comparative analysis is carried out between seven secure software development methodologies, under an international standard Common Criteria (ISO / IEC 15408). Throughout this study three chapters are developed, which allow describing concepts, antecedents, characteristics and finally make a comparative analysis between secure software development methodologies and as an object of study, it is observed that the secure development methodology, which complies With most of the established criteria, within the international standard it is the Security Development Lifecycle SDL methodology. The research allows to conclude that there are criteria that so far none of the methodologies have present them within their phases, therefore, it is considered that these aspects can be taken into account for future research.

INTRODUCCIÓN

La ingeniería de software se enfoca en establecer procesos que permitan ser controlarlos a través de buenas prácticas, con el objeto de construir aplicaciones con calidad que garanticen una buena imagen hacia el usuario final, en términos de funcionalidad del producto. Las metodologías de desarrollo se han enfatizado en construir un producto que cumpla básicamente con las necesidades de la lógica del negocio, dejando de un lado un elemento que últimamente ha tomado relevancia como es la seguridad.

Es muy común identificar que cuando se despliegan los proyectos en producción, se proceda a la corrección de errores, de los cuales se hubieran podido resolver en las fases de desarrollo, esto conlleva ciclos infinitos de revisiones en cuanto al diseño, ajustes, actualizaciones, generando tiempos de retraso, donde la gran parte de estas entregas se ven afectadas contractualmente.

Cuando se detecta que los errores están relacionados con aspectos puntales de desarrollo, se deben establecer una serie de estrategias de desarrollo de software seguro, que se encuentran descritas dentro de algunas metodologías, es importante tener en cuenta que las empresas de desarrollo, dentro de sus equipos cuenten con profesionales que sean capaces de desarrollar en menor tiempo proyectos con el objetivo de obtener productos que cumpla con los lineamientos establecidos, independiente de su metodología de desarrollo seguro que apliquen.

En la presente monografía se realizará un análisis comparativo entre siete metodologías de desarrollo seguro, donde se definirán unos criterios con base en el estándar ISO-15408 Common Criteria (CC), con el fin de identificar cuáles de estas metodologías cumple con los lineamientos descritos. Al finalizar en esta monografía, se realizará un manual de buenas prácticas de desarrollo seguro, de acuerdo con los lineamientos establecidos dentro de estas metodologías objeto de estudio.

1. DEFINICIÓN DEL PROBLEMA

1.1 ANTECEDENTES DEL PROBLEMA

Hoy en día se encuentran diferentes metodologías de desarrollo para software seguro tales como SDL[1], CBYC[2], Oracle software Security Assurance, CLASP[3], Team Software Process Secure[4], Software Assurance Maturity Model[5], Building Security in Maturity Model[6], Agile Devlopment Using Microsoft Security Development Lifecycle[7], entre otras, de las cuales se encuentran clasificadas dentro metodologías tradicionales y agiles. Desde el 2001, el ciclo de vida del desarrollo ha estado en constante cambio ya que las casas de software dentro sus procesos han venido implementando diversas metodologías agiles[8]; por consiguiente, el desarrollo de software viene en constante transición de metodologías[9]. Por lo anterior, es necesario entender que los participantes dentro del ciclo de vida de desarrollo de software conozcan las metodologías para el desarrollo seguro, decidiendo cuales cumplen con las expectativas y necesidades del negocio.

---

[1] POTTER. B.Microsoft SDL Threat Modelling.network Security [En línea].2009,Vol.2009Nro.1.[Consultado 27 de marzo 2020].Disponible en: https://sci-hub.tw/10.1016/s1353-4858(09)70008-x

[2] HALL,A,CHAPMAN,R.Correctness by construction: developing a commercial secure system.IEEE Software [en linea].2002,Vol.19Nro.1.[Consultado 15 de diciembre 2019].Disponible en:https://doi.org/10.1109/52.976937.

[3] GREGOIRE.On the Secure Software Development Process: CLASP and SDL Compared.Third International Workshop on Software Engineering for Secure Systems[en linea].2007,Vol.1Nro.1.[Consultado 11 de noviembre 2019].Disponible en: https://doi.org/10.1109/SESS.2007.7.

[4] NOOPUR,D y MULLANEY,J.The Team Software Process (TSP) in Practice: A Summary of Recent Results.IEEE Software [en linea].2003,Vol.2Nro.1.[Consultado 11 de noviembre 2019].Disponible en:https://doi.org/10.1184/R1/6585302.v1

[5]BURNSTEIN,I,SUWANASSART,Y CARLSON,R.Developing a Testing Maturity Model for software test process evaluation and improvement.Proceedings International Test Conference [en linea].1996,Vol.5819Nro.89.[Consultado 11 de noviembre 2019].Disponible en:https://doi.org/10.1109/TEST.1996.557106

[6] MCGRAW, G. Building Security In Maturity Model, s/f, 34.

[7] BACA,D.Y Bengt,C.Agile Development with Security Engineering Activities.[En linea].3°.ed.(Proceeding of the 2nd Workshop on Software Engineering for Sensor Network Applications;nro.11)USA:Edimat,2011.[Citado el 11 de diciembre del 2019].Disponible en: https://doi.org/10.1145/1987875.1987900.

[8] CADAVID,A.Revisión de metodologías ágiles para el desarrollo de software.Prospectiva [en linea].2013,Vol.11Nro.2.[Consultado 20 de noviembre de 2019].Disponible en:https://doi.org/10.15665/rp.v11i2.36.

[9] SALO,O,ABRAHAMSSON,P.Agile Methods in European Embedded Software Development Organisations: A Survey on the Actual Use and Usefulness of Extreme Programming and Scrum. IET Software [en linea].2008,Vol.2Nro.1.[Consultado 20 de diciembre 2019].Disponible en:https://doi.org/10.1049/iet-sen:20070038.

Investigación realizada por Hernández[10], donde el investigador menciona que uno de los componentes más costosos cuando sucede un ataque virtual, es la perdida de información ya que representa un 43% de los costos, debido que estos ataques son ocasionados por malware creados desde la web han hecho que mayoría de las compañías destinen recursos millonarios[11] en seguridad de la información, la investigación concluye que cada día se crean aproximadamente 230.000 malwares.

Investigación realizada por Campos y Rolando[12], identifico que un porcentaje alto de los ingenieros, no conocen de metodologías de desarrollo seguro ni mucho menos que las apliquen, debido que muchas compañías no cuentan con políticas de seguridad, es necesario reforzar el conocimiento de los ingenieros, donde se establezca dentro de sus procesos de desarrollo buenas prácticas de seguridad y estas sean implementadas dentro de las fases de desarrollo.

Como aspecto fundamental se ve la necesidad de dar a conocer al área de la ingeniería la filosofía de la seguridad[13] a todos los involucrados ya que es necesario que se concienticen de la importancia de aplicar seguridad en cada etapa del desarrollo. Dentro de este proceso es de vital importancia que se involucre todos los roles técnicos[14] que hacen parte dentro de los equipos de ingeniería.

Existe una preocupación latente por parte de los ingenieros desarrolladores, respecto al riesgo en la transacción de información en diferentes dispositivos a los que hoy se tienen acceso aumentando el riesgo, debido que hay un gran número de personas que manipulan esta información durante este proceso pueden que surjan errores técnicos y desencadenen en una vulnerabilidad[15].

Expuesto lo anterior, se evidencia un desconocimiento por parte del área de ingeniería de software sobre metodologías existentes para el desarrollo de software seguro, tomando como base los elementos más importantes como es la estrategia

---

[10] HERNANDEZ,Antonio.Sistema para la detección de ataques PHISHING utilizando correo electrónico.Revista Telemática 17.2019,nro.2.pp.60-70.ISSN 0120-3916

[11]SUKHAI,N.Hacking and cybercrime.Kennesaw, Georgia: Association for Computing Machinery [en linea].2004,Vol.19Nro.1.[Consultado 11 de noviembre 2019].Disponible en: https://doi.org/10.1145/1059524.1059553.

[12]CAMPOS,S y ROLANDO,J.Implementación de la metodología Six Sigma SDLC para la mejora de la calidad del proceso de desarrollo web,En: Repositorio UnaAcCr.1,No.1.2015.[En línea].Recuperado en 2019-94-140.Disponible en:https://repositorio.una.ac.cr/handle/11056/10550

[13] HANSSON,S.Risk and Safety in Technology,En: Philosophy of Technology and Engineering Sciences.1,No.1.2009.[En línea].Recuperado en 2019-1069-1102.Disponible en: https://doi.org/10.1016/B978-0-444-51667-1.50043-4.

[14] DE WIN,B.On the Secure Software Development Process: CLASP, SDL and Touchpoints Compared,En: Information and Software Technology.51,No.7.2009.[En línea].Recuperado en 2019-71-1152.Disponible en: https://doi.org/10.1016/j.infsof.2008.01.010.

[15] RODRIGUEZ, G,. Es seguro el uso del software en el intercambio de información bancaria.IEEE Software [en línea].2018,Vol.19Nro.75.[Consultado 28 de diciembre 2019].Disponible en:https://repository.unad.edu.co/bitstream/handle/10596/23770/52032204.pdf?sequence=5&isAllowed0=y.

de trabajo, entorno de aplicación y tiempo invertido, entre otras características que ayuden a tomar una decisión sobre la metodología a utilizar que encaje con la realidad de su empresa. Por consiguiente, surge la necesidad de identificar las buenas prácticas o lineamientos que se deben implementar bajo una metodología de software seguro, teniendo en cuenta las necesidades de la lógica de negocio.

1.2 FORMULACIÓN DEL PROBLEMA

¿Porque es importante hacer un análisis comparativo entre las metodologías de desarrollo seguro?

2 JUSTIFICACIÓN

El desarrollo de software seguro día tras día se ha convertido en un asunto primordial en gran parte de las compañías de software, debido al incremento de ataques en diferentes sistemas financieros, bancarios, públicos etc. Donde se ha convertido en un dolor de cabeza especialmente dentro del equipo de ingeniera de dichas compañías, dando lugar a que sean vulnerados de forma sencilla, por tal motivo es necesario aplicar metodologías de desarrollo para software seguro en cada una de las fases de desarrollo. Es importante garantizar que la seguridad sea un factor esencial dentro del proceso de desarrollo.

Según Microsoft[16], Colombia ocupa el tercer puesto en Latinoamérica con el mayor índice de vulnerabilidades. Estas cifras son avaladas por IDC[17], los ciberataques han aumentado entre un 30% y el 40% en América Latina. De acuerdo con la investigación anterior, este índice de vulnerabilidad se debe a que las aplicaciones no cuentan con el nivel de seguridad adecuado ya que se evidencia una ausencia de buenas prácticas contempladas dentro de las metodologías para desarrollo de software seguro.

En la investigación hecha por Herzog[18], se proyecta que para el 2021 el daño por ciberataques llegara a los $6 trillones de dólares anuales, por tal motivo es necesario que las compañías inicien a tomar medidas al respecto. Se puede inferir que el factor que causa que un gran número de ingenieros, analistas, testers y gerentes de proyecto no aplican metodologías de desarrollo seguro porque se ven afectados por el tiempo, ya que muchas compañías tienen como política entregar un desarrollo en el menor tiempo posible contemplado bajo unas cláusulas de contrato, donde se definen unos tiempos de entrega de un producto parcial o final.

Expuesto lo anterior es importante tomar medidas dentro de los equipos de ingeniería, el riesgo que se corre si se desarrolla un producto que no cumpla con los requisitos mínimos de seguridad, Por consiguiente, existen diversas soluciones para mitigar este riesgo implementando una metodología para el desarrollo de software seguro. En la actualidad existen diferentes metodologías para el desarrollo de software seguro; por lo tanto, para muchas compañías se convierte un tema dispendioso estudiar y aplicar una metodología que se acople a las expectativas y necesidades del negocio.

Por consiguiente, en esta monografía se realizará el estudio y diseño de un análisis comparativo entre las diferentes metodologías de acuerdo con el estándar ISO-15408, denominado Common Criteria (CC), establecido internacionalmente, con el fin de identificar, metodologías que cumplen con los objetivos corporativos de cada

[16] BALICA,F,WRIGHT,G,Y MEULEN,F.A Flood Vulnerability Index for Coastal Cities and Its Use in Assessing Climate Change Impacts. Natural Hazards [en linea].2012,Vol.64Nro.1.[Consultado 28 de diciembre 2019].Disponible en: https://doi.org/10.1007/s11069-012-0234-1.

[17] IDC COLOMBIA.[sitio web].Bogotá:IDC,Analiza el futuro.[Consulta: 16 de febrero 2020].Disponible en:http://www.idccolombia.com.co/.

[18] HERZOG,Felix.Straftaten im Internet, Computerkriminalitat und die Cybercrime Convention.Revista política criminal 4.2019,nro.8.pp.475-84.ISSN 0120-3916

empresa. Al aplicar metodologías para el desarrollo de software seguro beneficiará a las compañías y a sus colaboradores, ya que desarrollaran un producto que cumplirá con los estándares normativos contemplados en la ISO 27001[19] y ISO-15408, permitiendo una reducción de costos, la mitigación de errores de calidad y ataques ocasionados por malware.

[19] ALMEIDA,G.Legal Rules and Information Security Technical Standards: Possible Approach for Filling in the Blanks of Cybercrime Legislation. SSRN Electronic Journal [en línea].2011,Vol.2Nro.1.[Consultado 20 de febrero de 2020].Disponible en: https://doi.org/10.2139/ssrn.1742962.

3 OBJETIVOS

3.1 OBJETIVOS GENERAL

Realizar un análisis comparativo entre las metodologías para el desarrollo de software bajo el estándar Common Criteria (ISO/IEC 15408).

3.2 OBJETIVOS ESPECÍFICOS

- Identificar las metodologías que garanticen el cumplimiento de los criterios de desarrollo seguro de acuerdo con el estándar Common Criteria (ISO/IEC 15408).

- Realizar un análisis comparativo entre las metodologías de desarrollo seguro bajo el estándar Common Criteria (ISO/IEC 15408).

- Realizar un manual para la aplicación de metodologías de desarrollo de software seguro bajo el estándar Common Criteria (ISO/IEC 15408) y tamaño de grupos de desarrollo.

4 MARCO REFERENCIAL

Para abordar el contexto de esta monografía; en este capítulo se ha descrito un marco de referencia con diferentes estudios o teorías que se han realizado en los últimos cinco años en el tema de interés, como es la importancia de implementar metodologías para el desarrollo seguro especialmente en los equipos de desarrollo.

Debido a que en el mundo actual los ataques a compañías ya sean grandes o pequeñas vienen en un crecimiento exponencial.

4.1 MARCO TEÓRICO

El grado de exigencia de seguridad de cada uno de sus componentes, cómo los diferentes niveles exigidos en los productos de desarrollo dentro del ciclo de vida, es necesario que se pueda abordar en la fase de análisis y evaluación de riesgos, la aplicación de una metodología de desarrollo seguro bajo un estándar internacional[20].

Por lo tanto, es necesario referenciar las siguientes teorías para soportar esta monografía.

4.1.1 Estándar internacional ISO / IEC 15408 Common Criteria (CC).

Según los autores Mellano y Fernández, lo definen como:
"Un estándar internacional (ISO / IEC 15408), para seguridad informática donde su propósito es, permitir que los usuarios especifiquen sus requisitos de seguridad, para permitir a los desarrolladores definan los atributos de seguridad de sus productos y finalmente los evaluadores determinen si los productos realmente cumplen con sus reclamos"[21].

De acuerdo con la anterior teoría se identifica plenamente con la monografía, debido que se va a realizar un análisis comparativo, entre siete metodologías para el desarrollo de software seguro, por lo cual se requiere definir unos criterios que se deben definir bajo un estándar internacional, para este estudio se seleccionó el estándar internacional ISO / IEC 15408 Common Criteria (CC). A través de este

[20]MITRA, R. Security Level Identification and Secure Software Design of Safety Critical Embedded Systems: Methodologies and Process, En: Incose International Symposium.27,No.1.2017.[En línea].Recuperado en 2020-1300-1313.Disponible en:https://doi.org/10.1002/j.2334-5837.2017.00429.x.
[21] Ibíd., p.23

estándar, se presenta un proceso centrado en criterios comunes y basados en la reutilización de los requisitos de seguridad en las primeras etapas del desarrollo de software de una manera, sistemática e intuitiva, proporcionando un repositorio de recursos de seguridad e integrando los criterios comunes en el ciclo de vida del software.

4.1.2 Desarrollo seguro.

Según los autores Brown y Paller se define como:
"Una necesidad en el diseño y desarrollo de software, si se realiza un correcto desarrollo de software, se puede evitar fallos de seguridad que pueden significar grandes pérdidas de tiempo, información, dinero y estabilidad de este. Al igual que en el pasado se desarrollaron ciclos de software para mejorar la detección de bugs y diseño de aplicaciones, actualmente se está incorporando la seguridad a estos ciclos de desarrollo"[22].
De acuerdo con lo anterior expuesto, la teoría se identifica plenamente con la monografía, porque es necesario divulgar que la seguridad también se debe enfocar en el desarrollo de software, con el fin de poder garantizar la seguridad desde las etapas más tempranas del ciclo de vida de desarrollo, creando un producto software con los estándares de seguridad mínimos, que sea capaz de resistir ataques, alta resiliencia y mitigar el daño cuando es ocasionado por los ataques. Por consiguiente, nace la necesidad de implementar prácticas de seguridad en cada fase de desarrollo de acuerdo con una metodología de desarrollo seguro.

4.1.3 Software Seguro**.**

Según Brito**,** se define como: "Proceso de desarrollo donde se construye una aplicación que pueda resistir o sostenerse ante ataques, y además se pueda recuperar rápidamente y mitigar el daño causado por la explotación de vulnerabilidades que no puedan ser eliminadas o resistidas" [23].
De acuerdo con la anterior teoría se identifica con la monografía, dado que corresponde a la idea central de la investigación para que este proceso, es necesario contar con un grupo de ingenieros desarrolladores comprometidos y capacitados que apliquen las buenas prácticas de seguridad definidas dentro de una metodología de desarrollo seguro.

4.1.4 El ciclo de vida de desarrollo de software seguro**.**

[22] BROWN,M y PALLER,A.Secure Software Development: Why the Development World Awoke to the Challenge, En: Information Security Technical Report.13,No.1.2008.[En línea].Recuperado en 2020-40-43.Disponible en: https://doi.org/10.1016/j.istr.2008.03.001.
[23]BRITO, C.Metodologías para desarrollar software seguro, En: Universidad Autónoma de Zacatecas .2,No.3.2013.[En línea].Recuperado en 2020-2-18.Disponible en: https://www.redalyc.org/pdf/5122/512251564005.pdf.

Según Futcher y Rossouw lo definen como: “Un conjunto de principios de diseño y buenas prácticas a implantar en el ciclo de vida de desarrollo de software, ya que permite detectar, prevenir y corregir los defectos de seguridad en el desarrollo y adquisición de aplicaciones, de forma que se obtenga software de confianza y robusto frente a taques maliciosos, que realice solo las funciones para las que fue diseñado, que esté libre de vulnerabilidades, ya sean intencionalmente diseñadas o accidentalmente insertadas durante su ciclo de vida y se asegure su integridad, disponibilidad y confidencialidad”[24].
De acuerdo con lo anterior expuesto, la teoría se identifica plenamente con la monografía, porque es necesario identificar a detalle cada una de las faces que hacen parte del ciclo de vida con el fin de identificar que practica se puede alinear de acuerdo con la metodología de desarrollo seguro que se aplique.

4.1.5. Pyme**.** Según los autores Hogan, Smith y Thomas se define como: “Un término que se enfoca de cómo se debe clasificar las compañías de acuerdo con la cantidad de empelados teniendo como referencia la región o país como lo establezca dentro de sus leyes”[25].
De acuerdo con lo anterior expuesto, la teoría permite identificar a nivel del mundo como se típica las empresas es decir cómo se clasifican como grandes, medianas y pequeñas de acuerdo con el número de empleados.

4.2 MARCO CONCEPTUAL

4.2.1. ESTADO DE ARTE

Es necesario saber cómo está el país frente a los otros países en el mundo, en el tema de desarrollo para software seguro, en la siguiente investigación realizada por Dávila[26], hace un énfasis que Colombia frente a la seguridad de software le falta mucho para competir con los países potencia en cuanto seguridad como por ejemplo Estados Unidos, esta nación se considera dependiente de las tecnologías de la información, siendo uno de los focos centrales la seguridad al momento de innovar soluciones, dado que en últimos años se ha posicionado en uno de los

[24] FUTCHER,L y VON SOLMS,R.Guidelines for secure software development, En: Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries.8,No.1.2008.[En línea].Recuperado en 2020-56-65.Disponible en: https://doi.org/10.1145/1456659.1456667.
[25]HOGAN, J.; SMITH,G y THOMAS,R.The Real World Software Proces, En: Ninth Asia-Pacific Software Engineering Conference.29,No.2.200.[En línea].Recuperado en 2020-366-75.Disponible en: https://doi.org/10.1109/APSEC.2002.1183006.
[26] SGUERRA,Davila. Diseño de software seguro.4 Ed. Bogotá D.C:2006.8p.(acistente;nro 43).ISBN-41-0131-5.

mercados más fuertes en el mundo en cuanto el desarrollo seguro, así lo demuestra el último reporte de la segunda cumbre del Centro Nacional de Software[27].

Puesto a lo anterior se hace un llamado a todas las casas de software que tomen medidas al momento de desarrollar soluciones, que brinden servicio a entidades públicas y privadas que dentro de sus procesos se inicien a implementar alguna metodología para el desarrollo de software seguro bajo un estándar internacional [28]. Continuando con el estado actual de Colombia frente al tema de la seguridad en campo del desarrollo, se menciona esta otra investigación realizada por Poggi[29], donde el investigador menciona, que en el año 2019, el 53% de los atacantes se aprovechan de las vulnerabilidades de software como una ventaja de acceso a los sistemas de información, debido al auge del mercado del desarrollo de software, han expuesto nuevas metodologías de desarrollo que han modificado las fases de desarrollo[30] y que han ocasionado que se dejen esas ventanas de acceso a los sistemas de información.
En la siguiente investigación realizada por Aljawarneh[31], se enfatiza que la seguridad se considera uno de los requisitos no funcionales que tienen un efecto significativo en el diseño arquitectónico del software como servicio, en esta investigación manifiesta que el 70% de los ingenieros desarrolladores no conocen de metodologías para el desarrollo de software seguro y esto presenta un desafío significativo para el equipo de desarrollo de software para lidiar con la seguridad en la etapa de desarrollo, el autor[32], indico que los resultados de la evaluación mostraron la aparición de un número significativo de vulnerabilidades de seguridad en las primeras etapas del ciclo de vida de desarrollo de software.

Los investigadores Nivia y Cortes[33], mencionan que la seguridad en la etapa de desarrollo se convirtió en una necesidad que debe resolverse desde el área de la

---

[27] National Security and Competitiveness - ProQuest.[sitio web].Bogotá:IDC,A Study on How Cyber Economic Espionage Affects .[Consulta: 22 de marzo 2020].Disponible en: https://search.proquest.com/openview/7554ba20146515e188f6a371c1e1bea5/1?pq-origsite=gscholar&cbl=18750&diss=y.

[28] DAVILA, op.cit,p.10

[29]VELASCO,C,Y HARDANY,E.El mundo cibernético y los delitos informáticos. Repositorio Institucional USC [en linea].2019,Vol.64Nro.1.[Consultado 22 de marzo 2020].Disponible en:https://repository.usc.edu.co/handle/20.500.12421/510.

[30] AVELT,M ,ANUSHA,P y LOELLA,M.Secure SDLC for IoT Based Health Monitor: A Summary of Recent Results.Second International Conference on Electronics [en linea].2018,Vol.2Nro.1.[Consultado 30 de marzo 2020].Disponible en: https://doi.org/10.1109/ICECA.2018.8474668

[31] SHANDI,A,LAWNEH,A,Y JARADAT,R.Cloud Security Engineering: Early Stages of SDLC, Future Generation Computer Systems [en linea].2017,Vol.74Nro.1.[Consultado 25 de marzo 2020].Disponible en: https://doi.org/10.1016/j.future.2016.10.005.

[32] ALJAWARNEH, op. cit, p.90

[33] NIVIA,R,CORTES,P,Y ROJAS,E.Implementation Phase Methodology for the Development of Safe Code in the Information Systems of the Ministry of Housing ,en Computational Science and Its Applications – ICCSA 2018, ed. Osvaldo Gervasi et al [en linea].201,Vol.74Nro.1.[Consultado 25 de marzo 2020].Disponible en: https://doi.org/10.1007/978-3-319-95165-2_3.

tecnología. Dado que la información es el activo más importante de cualquier organización, es importante en los sistemas de información generar altos niveles de seguridad, integridad y confiabilidad. Por consiguiente, es importante aplicar una metodología para el desarrollo de código seguro con los procedimientos e indicaciones necesarios para evitar posibles ataques a la seguridad de la información y con el objetivo de cubrir la fase de desarrollo en el proceso de creación de sistemas.

En la siguiente encuesta realizada por Galov[34], realizada a más de 1.300 profesionales de TI se descubrió que 56% de las organizaciones identificaron en la técnica phishing[35] se presenta el mayor riesgo de seguridad informática, debido que esta técnica es usada por delincuentes cibernéticos para estafar con el fin de obtener información sensible de forma fraudulenta.
De acuerdo con la investigación realizada por Mwangi, Shedden y Masupe[36], Se realiza una estimación donde se identifica que actualmente, el 50% de las vulnerabilidades de los sistemas tienen su origen desde el diseño del producto.
Debido al momento de implementar se logra identificar fallas conocidas como Bugs, por consiguiente[37], menciona que el origen se presenta desde la codificación. Es necesario implementar una metodología que garantice un desarrollo con calidad.

Los investigadores Medina y Mesías[38],concluyeron que al aplicar buenas prácticas desarrollo seguro, definidas dentro de una metodología, basada en un estándar internacional permite aumentar las cifras de productividad, bajan los costos, recupera la confiabilidad de los clientes, reducción de tiempos de desarrollo, de manera eficiente y rápida, por lo anterior los investigadores[39], destacan que es un desafío identificar metodologías de desarrollo seguro cumpla con las necesidades y expectativas de las compañías, es necesario realizar un análisis previo donde se evalué o se identifique las necesidades de estas y así no incurrir en costos sobredimensionados.

---

[34] SLUPSK,J,TADDEO,M.Generative Metaphors in Cybersecurity Governance, en The Yearbook of the Digital Ethics Lab, ed. Christopher Burr y Silvia Milano, Digital Ethics Lab Yearbook [en linea].2020,Vol11.Nro.30.[Consultado 26 de marzo 2020].Disponible en:https://doi.org/10.1007/978-3-030-29145-7_2.

[35] INFOSPYWARE. [sitio web].Bogotá:CEO,¿Qué es el Phishing?.[Consulta: 9 de abril 2020].Disponible en:https://www.infospyware.com/articulos/que-es-el-phishing/.

[36] MWANGI,E,MASUPE,S,Y GASENNELWE,M.Formal Specification for Internet of Things Malware,en 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE) (2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)[en linea].2018,Vol.144Nro.49.[Consultado 25 de marzo 2020].Disponible en: https://doi.org/10.1109/iCCECOME.2018.8659285

[37] Lbid.,p12

[38] MEDINA,Nohem y MESIAS,Wilmer.ESCOGER UNA METODOLOGÍA PARA DESARROLLAR SOFTWARE, DIFÍCIL DECISIÓN.Revista Educación en Ingeniería 10.2015,nro.20.pp.98-109.ISSN 10.26507

[39] MEDINA, MESIAS,op.cit,p.12

Puesto a lo anterior en la siguiente investigación realizada por Hudaib, AlShraideh, Surakhi y Khanafseh[40], permite mencionar que existe muchos conceptos específicos de seguridad, de los cuales se deben determinar los requisitos durante el ciclo de vida de desarrollo de software para ofrecer un entorno sólido y seguro. Se analizaron varios procesos y metodologías existentes necesarias para desarrollar soluciones seguras de software, basado en los trabajos publicados relacionados, donde se comienza presentando el desarrollo de software seguro más relevante, se propone realizar una comparación entre siete metodologías de desarrollo seguro, de acuerdo a unos criterios ya definidos bajo el estándar Common Criteria (ISO/IEC 15408), teniendo en cuenta que estos criterios se deben ejecutar en cada fase de desarrollo[41]. Cuando se selecciona algunas de estas metodologías se convierte en un desafío, al momento de tomar una decisión debido que en esta investigación se manifestó un interrogante, como es saber ¿cuál usar sin saber la diferencia entre ellas?, debido a este interrogante se precedió a realizar esta investigación mencionan los autores[42], donde se ofrece una breve descripción de cada una de las metodologías con el fin de realizar una comparación entre estas metodologías de acuerdo con unos criterios. Los resultados muestran que la mayoría de ellas no incluyen las actividades de seguridad en todas las fases de desarrollo, los procesos, SDL[43] y CLASP[44] cubren más aspectos de seguridad desarrollo de software con la definición de una lista de ventajas y desventajas[45].

Apoyando al proceso de desarrollo se permite relacionar la siguiente investigación realizada por Silva, Noel, Matalonga, Astudillo, Gatica y Marquez[46], identificaron que dentro de las aplicaciones de software existen, una variedad de procesos para lograr un software seguro dentro de las aplicaciones. Sin embargo, la mayoría de estos carecen de evidencia empírica de su aplicación e impacto en la construcción de sistemas de software seguros. Se han realizado dos estudios de mapeo sistemático

[40] SURAKHI,O,A Survey on Design Methods for Secure Software Development. INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY [en linea].2017,Vol.16Nro.7.[Consultado 22 de marzo 2020].Disponible en:https://doi.org/10.24297/ijct.v16i7.6467.

[41] Ibid.,p12.

[42] Lbid.,p23.

[43]POTTER. B.Microsoft SDL Threat Modelling.network Security [en línea].2009,Vol.2009Nro.1.[Consultado 27 de marzo 2020].Disponible en: https://sci-hub.tw/10.1016/s1353-4858(09)70008-x

[44] DE WIN, B.; ,SCANDARIATO,R.;BUYENS,K,.;GREGORIE,J y WOUNTER,J .On the secure software development process CLASP, Network Security.51,No.7.2009.[En línea].Recuperado en 2009-1152-1171.Disponible en:https://www.sciencedirect.com/science/article/abs/pii/S0950584908000281

[45] Ibid.,p12

[46] SILVA, P.; ,MATALONGA,S.;NOEL,R y ASTUDILLO,H .Methodologies to Identify and Mitigate Security Threats in Software Development: Two Systematic Mapping Studies, Researchdate.19,No.3.2016.[En línea].Recuperado en 2016-10-19153.Disponible en:https://www.researchgate.net/publication/311424013_Methodologies_to_Identify_and_Mitigate_Security_Threats_in_Software_Development_Two_Systematic_Mapping_Studies

(SM)[47], realizado para cubrir las iniciativas existentes para la identificación y mitigación de la seguridad.

Se realizaron varios estudios con el objetivo de explorar decisiones de diseño de software para desarrollar software seguro sistemas, inicialmente se adoptó un enfoque experimental para comparar tácticas de seguridad de software y seguridad en cuanto patrones de diseño, y surgieron varias preguntas al intentar operacionalizar la identificación y mitigación de amenazas de seguridad[48].
Fue necesario investigar que implicación se presenta al realizar una evaluación experimental de metodologías de software seguras utilizando patrones de acuerdo con la investigación realizada por Fernández y Astudillo[49], donde los autores mencionan la importancia de implementar metodologías que apliquen, patrones de seguridad debido que estos patrones se consideran artefactos útiles para construir sistemas seguros. Una razón es que casi no hay evaluaciones empíricas de su valor para construir sistemas seguros, menciona los autores[50], que revisaron este tema y consideraron que los nuevos experimentos podrían ser útiles para demostrar el valor de los patrones de seguridad, en cuanto la utilidad y el empleo dentro de las metodologías de desarrollo seguro.

Se encuentra la siguiente investigación por Mohammad y Alqatawna[51], donde realizó un análisis comparativo entre dos metodologías enfocadas para el desarrollo de software seguro, SDLC y OWASP, donde esta investigación está restringido a dos representantes de vanguardia, a saber, el ciclo de vida de desarrollo de seguridad (SDL) de Microsoft[52] y OWASP[53]. Las características generales de los procesos tienen se ha descrito, así como las diferencias específicas en las diversas fases de desarrollo, desde la perspectiva de las actividades del proceso. Los

[47] RUSSO, J y SOLARI, M .Estudio de Mapeo Sistemático sobre Arquitecturas de Software para Big Data, Researchdate.19,No.3.2017.[En línea].Recuperado en 2017-10-357.Disponible en:https://www.researchgate.net/publication/331001020_Estudio_de_Mapeo_Sistematico_sobre_Arquitecturas_de_Software_para_Big_Data

[48] Ibid.,p13

[49]FERNANDEZ, E y ASTUDILLO,H.Experimental evaluation of secure software methodologies using patterns, En:Acm dl Digital Library.16,No.11.2016.[En línea].Recuperado en 2020-1-7.Disponible en:https://dl.acm.org/doi/10.5555/3124362.3124368

[50] Ibíd., p.14

[51] MOHAMMAD, A.; ALQATAWNA, J y MOHAMMAD,A.Secure software engineering: Evaluation of emerging trends, En: 8th International Conference on Information Technology (ICIT).8,No.11.2017.[En línea].Recuperado en 2017-814-18.Disponible en: https://doi.org/10.1109/ICITECH.2017.8079952.

[52]TONDEL, I.Understanding challenges to adoption of the Microsoft elevation of privilege game, En: Understanding challenges to adoption of the Microsoft elevation of privilege game.18,No.11.2018.[En línea].Recuperado en 2018-1-10.Disponible en: https://doi.org/10.1145/3190619.3190633.

[53] WICHERS, D.OWASP Top-10 2017, En: Owasp the Open Web Application Securty Project .1,No.10.2017.[En línea].Recuperado en 2017 -1-43.Disponible en: https://upload.wikimedia.org/wikipedia/mediawiki/archive/e/e9/20180111214627%21OWASP_Top-10_2017_-_Presentation.pdf.

autores[54] menciona que SDL ofrece un proceso bien guiado dirigido a la seguridad como soporte calidad de software, mientras que CLASP[55] aborda la seguridad desde una perspectiva más amplia y se puede adaptar de manera flexible al entorno de desarrollo específico.

De acuerdo con el estudio realizado por Forrester[56], menciona que el 46% de las empresas que fueron encuestadas en su estudio, hacen énfasis en la necesidad de una metodología de desarrollo seguro propio, ya que al contar con una metodología propia es más sencillo adaptar al modelo de negocio de cada compañía. Es necesario tener en cuenta que la metodología seleccionada cumpla con los criterios que sean establecidos bajo un estándar internacional.
A continuación, se menciona una serie de investigaciones que se realizaron alrededor del estándar internacional estándar ISO/IEC 15408 Common Criteria.

En la siguiente investigación por Pino y López[57], se formuló y diseñó un modelo que establece pautas de evaluación enfocadas en la seguridad informática bajo un estándar internacional ISO/IEC 15408 Common Criteria, con el fin de lograr identificar los productos se les puede aplicar evaluaciones de seguridad.

Es necesario conocer más sobre el estándar internacional por consiguiente en la siguiente investigación Elbakyan[58] planteó una introducción técnica del estándar y su aplicación en cada uno de los criterios, dentro de una metodología para el desarrollo de software seguro, a través de un análisis enfocado a una evaluación por niveles que pueda proporcionar una alta eficiencia de tiempo, pero también puede proporcionar más justicia, más corrección y más precisión de resultados en la etapa de pruebas.

[54] MOHAMMAD,ALGATAWNA ,op,cit,p.14
[55] RAMACHANDRAN, M. Software Security Requirements Management as an Emerging Cloud Computing Service, En: International Journal of Information Management.36, No.4.2016.[En línea].Recuperado en 2020-580-90.Disponible en: https://doi.org/10.1016/j.ijinfomgt.2016.03.008.
[56] ENCK, W.A Study of Android Application Security, En: Systems and Internet Infrastructure Security.1,No.4.2011.[En línea].Recuperado en 2020-3-38.Disponible en: https://www.usenix.org/legacy/event/sec11/tech/slides/enck.pdf.
[57] CHAMORRO, J y PINO, F. Modelo para la evaluación en seguridad informática a productos software, basado en el estándar ISO/IEC 15408 Common Criteria, En: Sistemas y Telemática.9,No.19.2011.[En línea].Recuperado en 2020-69-92.Disponible en: https://doi.org/10.18046/syt.v9i19.1095.
[58] POTTI, O .;ILLIASHENKO,O y KOMIN,D.Advanced Security Assurance Case Based on ISO/IEC 15408, En: Theory and Engineering of Complex Systems and Dependability, ed. Wojciech Zamojski et al.365,No.11.2015.[En línea].Recuperado en 2020-391-401.Disponible en: https://doi.org/10.1007/978-3-319-19216-1_37.

Teniendo en cuenta los criterios que exige el estándar Common Criteria[59], descritos por Tallon[60], se identifica que al imponer el uso de metodologías basados en estándares de desarrollo reconocidas permiten hacer un alcance claro en cuanto la implementación del requisito ALC_TAT que produzca resultados consistentes y predecibles**.**

4.3 MARCO LEGAL

En Colombia hasta el momento no existen leyes o normas que rigen las casas de software, cuando se desarrolle software seguro, sin embargo, la Ley 1273 de enero de 2009[61], menciona unos artículos que describen las sanciones que se aplican a personas que realice algún tipo de acción delincuencial.

- Cuando se infringe el delito contemplado en el artículo (269B obstaculización ilegítima de sistema informático o red de telecomunicación): Aquel que obstaculice o impida el funcionamiento en cuanto la transferencia de información a través de sistema de dudosa procedencia, se le aplicara una pena en prisión de (48) a (96) meses y una multa de 100 a 1000 S.M.M.L.V.
- Cuando se infringe el delito contemplado en el artículo (269D daño informático): Aquel que realice una acción a través inyección de SQL elimine, actualice o inserte información, se le aplicara una pena en prisión de (48) a (96) meses y una multa de 100 a 1000 S.M.M.L.V.
- Cuando se infringe el delito contemplado en el artículo (Artículo 269E uso de software malicioso): Aquel que produzca, adquiera, introduzca o distribuya software malicioso dentro del territorio nacional, le repercutirá una pena en prisión de (48) a (96) meses y una multa de 100 a 1000 S.M.M.L.V.

[59] MELLANO, D .;FERNANDEZ,E y PIATTINI,M.A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems, En: Computer Standards & Interfaces.29,No.2.2007.[En línea].Recuperado en 2020-244-53.Disponible en: https://doi.org/10.1016/j.csi.2006.04.002

[60] TALLON, J. Common Criteria: Herramienta para el desarrollo seguro, En: Jtsec Beyond it Security.3,No.1.2018.[En línea].Recuperado en 2020-10-59.Disponible en: https://es.slideshare.net/javitallon/common-criteria-herramienta-para-el-desarrollo-seguro-97058338.

[61] GONZALEZ,D.La protección de información y los datos en el marco de la ley 1273 de 2009: un estudio del dato y la información como objeto material en el tipo penal hurto por medios informáticos, En: Universidad La Gran Colombia.1,No.1.2017.[En línea].Recuperado en 2020-5-78.Disponible en: http://repository.ugc.edu.co/handle/11396/4273.

5 ANÁLISIS DE METODOLOGÍAS PARA DESARROLLO SEGURO

5.1 IDENTIFICACIÓN DE LAS METODOLOGÍAS QUE GARANTICEN EL CUMPLIMIENTO DE LOS PROTOCOLOS DE SEGURIDAD EXIGIDOS CON BASE AL ESTÁNDAR INTERNACIONAL (ISO/IEC 15408)

A lo largo de este capítulo se identificará aquellas metodologías que cumplan con los criterios que exige el estándar, debido que se requiere identificar qué características se adaptan al estándar con el fin de realizar un análisis comparativo requerido para desarrollar cada uno de los objetivos propuestos en esta monografía.

5.1.1 Recopilación de información

5.1.2 Estándar Common Criteria ISO 15408 y desarrollo seguro

El estándar Common Criteria ISO 15408[62], se enfoca especialmente a la evaluación de software y hardware, con este estándar se podrá definir requisitos de seguridad de acuerdo, con los atributos de la seguridad como es la confidencialidad, disponibilidad, integridad, y la autenticación.

Este estándar en la actualidad se ha convertido muy importante para las empresas, se puedan certificar en el nivel de seguridad que ofrecen sus productos. De lado del usuario es interesante porque es más sencillo identificar qué nivel de seguridad puede pedirle al producto que compra[63].

A continuación, se menciona las principales características que hacen parte del estándar Common Criteria ISO 15408.

- Los certificados que se pueden obtener con el estándar cuentan con reconocimiento internacional.
- Los productos que se encuentren certificados con Common Criteria (CC), cuentan con un elemento diferenciador ya que aplican metodologías con evaluaciones sólidas[64].

---

[62] BARCELL, M. En: Universidad de Cadiz.2,No.3.2003.[En línea].Recuperado en 2020-2-18.Disponible en: http://www.mfbarcell.es/conferencias/Metodolog%C3%ADas%20de%20seguridad_2.pdf

[63] GEFEI, S.A supporting tool for creating and maintaining security targets according to ISO/IEC 15408, En: IEEE International Conference on Computer Science and Automation Engineering .3, No.3.2012.[En línea].Recuperado en 2020-745-49.Disponible en: https://doi.org/10.1109/ICSESS.2012.6269574.

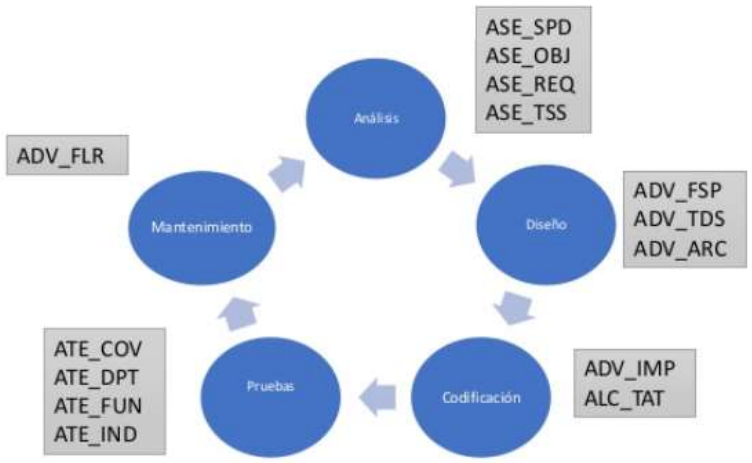
[64]MYUNG, S.A Design of Configuration Management Practices and CMPET in Common Criteria Based on Software Process Improvement Activity , En: Computational Science and Its Applications

Fases del estándar Common Criteria ISO 15408

A continuación, se describen las principales fases de desarrollo que intervienen al momento de crear un producto, bajo los requisitos definidos en el estándar Common Criteria ISO 15408[65].

En la Figura 1, se presenta un modelo que permitirá a los desarrolladores evaluar sus productos bajo un estándar a través de diferentes fases con el fin de realizar un análisis de riesgos.

Figura 1. Estándar Common Citeria ISO 15408



Fuente VETTERLING,Monica.Estándar Common Citeria ISO 15408,2001.[Imagen digital].Gráfica digital.(10,34x15.29cm)CM SIGSOFT Software Engineering Notes,Nueva York.Recuperado de:https://dl.acm.org/doi/10.1145/605466.605486#sec-ref el 22 de febrero de 2020.

ICCSA
.2,No.3.2004.[En línea].Recuperado en 2020-481-90.Disponible en: https://doi.org/10.1007/978-3-540-24707-4_59.

[65] Javier Tallón, “Common Criteria: Herramienta para el desarrollo seguro”, https://es.slideshare.net/javitallon/common-criteria-herramienta-para-el-desarrollo-seguro-97058338.

Fase de Análisis

- Requisito ASE_SPD: Se define como un problema de seguridad, debido que se requiere planear el análisis que permita dar una solución a un incidente, desde el enfoque de la seguridad[66].

En la Figura 2, se describe cada uno de los elementos en que se enfoca en esta etapa de análisis como son los activos a proteger como es la información.

Figura 2.Fase de análisis

Fuente: VETTERLING, Monica. Fase de análisis,2001. [Imagen digital]. Gráfica digital. (4.79x16.11cm) CM SIGSOFT Software Engineering Notes,Nueva York. Recuperado de: https://dl.acm.org/doi/10.1145/605466.605486#sec-ref el 22 de febrero de 2020.

- **Requisito ASE_REQ**: En esta fase se definen los requisitos funcionales de seguridad (SFRS), con el fin de que el producto cumpla con los objetivos de seguridad[67].

Fase de Diseño

- Especificación funcional (ADSV_FSP): Se realiza un análisis descriptivo de cada uno de los requisitos funcionales.
- Diseño del TOE (ADSV_TDS): Se define un diseño de cada uno de los componentes del TOE, a través de subsistemas y de módulos, de acuerdo con el nivel de evaluación.

[66] Shoichi Morimoto et al., "Formal Verification of Security Specifications with Common Criteria", en *Proceedings of the 2007 ACM Symposium on Applied Computing - SAC '07* (the 2007 ACM symposium, Seoul, Korea: ACM Press, 2007), 1506, https://doi.org/10.1145/1244002.1244325.

[67] HORIE, D .;MORIMOTO,S y CHENG,J.A Web User Interface of the Security Requirement Management Database Based on ISO/IEC 15408, En:Computational Science – ICCS 2006, ed. Vassil N.1,No.1.2006.[En línea].Recuperado en 2020-797-804.Disponible en: https://doi.org/10.1007/11758549_107.

- Arquitectura de seguridad (ADSV_ARC): Se describe y se define un diseño de seguridad desde el momento en que inicia a desarrollar el producto hasta que alcance un estado seguro.

Fase de codificación

- Requisito ALC_TAT**:** En este requisito se impone el uso de la herramientas y estándares de desarrollo seguro**.**
- Requisito ADV_IMP: Se obliga a presentar dentro del código fuente la respectiva evaluación de seguridad.

Fase de pruebas

- Requisito ATE_FUN: Se elabora un plan de pruebas con el fin de obtener resultados con base de unos escenarios previamente definidos.
- Requisito ATE_IND: Los evaluadores realizaran varias repeticiones de las pruebas, donde se proporciona un producto con base de un entorno adecuado.
- Requisito ATE_COV: Permite identificar el comportamiento de las interfases respecto a la funcionalidad de la seguridad.
- Requisito ATE_CPT: Permite identificar el comportamiento de los subsistemas y módulos respecto a la funcionalidad de la seguridad.

Fase de Mantenimiento

- Requisito ALC_FLR: Consiste en corregir fallos, establece una serie de pasos para solucionar un problema.

5.1.2.1 Metodologías para el desarrollo seguro

Es necesario hacer uso de herramientas donde, estas se deben aplicar en las fases de desarrollo para instrumentarlo en procesos de modelado que se enfoquen a todos los equipos de desarrollo, debido que la gran mayoría de veces los profesionales no son muy expertos en seguridad.

A continuación, se describen que metodologías de desarrollo seguro existen, de las cuales hay unas que son muy conocidas:

5.1.2.2 Metodología security development lifecycle (SDL)

Microsoft Security Development Lifecycle[68] , Esta metodología se puede aplicar dentro del desarrollo de software, esta metodología podrá soportar ataques de seguridad[69], este proceso de desarrollo de software de Microsoft, contempla dentro de su proceso actividades que se enfoca, en cuanto la seguridad en cada entregable que se hacen para cumplir cada fase del desarrollo de software.

Security Development Lifecycle (SDL)[70], se diseñó con Trustworthy Computing (TwC) hacia el año 2002. Donde conto con muchos, grupos de desarrollo de software en Microsoft que comenzaron a encontrar formas para mejorar la seguridad del código existente.

Microsoft SDL fue diseñado como una parte integral del proceso de desarrollo de software en Microsoft y publicado como una política obligatoria en 2004. El desarrollo, implementación y mejora constante de la SDL adoptada en Microsoft. Como resultado de esta política, el software es diseñado, desarrollado y probado para seguridad. El SDL de Microsoft ha madurado debido que es una metodología bien definida.

[68] Ibid.,p21
[69] MICROSOFT.[sitio web].Estados Unidos:SDLC.Microsoft SDL Progress Report.[Consulta: 4 de marzo 2020].Disponible en:https://www.microsoft.com/en-us/download/details.aspx?id=14107
[70]Ibid.,p21

En la Figura 3, Se describe cada una de las fases que se requiere para cumplir con la metodología de desarrollo seguro SDL.

Figura 3.Metodologia security development lifecycle (SDL)

Fuente: RUSCHER, Stéphane.Metodologia security development lifecycle (SDL),2011.[Imagen digital].Gráfica digital.(4.82x14.26cm),Les coulisses de la stratégie sécurité de Microsoft.Recuperado de:https://www.clubic.com/antivirus-securite-informatique/article-453036-2-windows-securite.html el 01 de marzo de 2020.

A continuación, se describe las fases desarrollo junto con las características

- **Primera etapa: Requisitos**

En esta fase se destaca los siguientes elementos que se deben tener en cuenta

- Seguridad y la privacidad: Se realiza un análisis de cada uno de los requisitos de que tengan que ver con la seguridad y privacidad dentro del entorno.

- Umbrales de calidad y límites de errores: Se proporciona una calificación al producto a través de una evaluación de vulnerabilidades.

- Evaluación de los riesgos de seguridad y privacidad: Este proceso es obligatorio, debido que se identifica los aspectos funcionales al producto a través de una serie de revisiones en cuanto el diseño de la seguridad con base en unos estándares (alto, medio, bajo).

- **Segunda etapa: Diseño**

- Requisitos: Se establece como se debe realizar la implementación de una manera que garantice la seguridad toda la funcionalidad del software.

- Modelo de riesgos: Se describen de manera estructurada las posibles implicaciones de seguridad que se pueda implementar en los diseños en el marco funcional.

- **Tercera etapa: Implementación**

- Usar herramientas aprobadas: Se debe hacer uso de compiladores, framework, que sean herramientas aprobadas a fin de identificar las nuevas actualizaciones enfocadas a seguridad.

- Prohibir funciones no seguras: En esta característica se debe verificar las funciones y API que se van a usar, si se logra identificar alguna irregularidad se debe prohibir las que consideran inseguras.

- Análisis estático: Se realiza una revisión en el código en cuanto la de seguridad que sea de una forma escalable bajo unas directrices.

- **Cuarta etapa: Comprobación**

- Análisis dinámico de los programas**:** se ejecuta un proceso AppVerifier[71], en conjunto con pruebas de exploración de vulnerabilidades.

- Pruebas de exploración de vulnerabilidades: Este proceso se enfoca en provocar una serie de errores, introduciendo una serie de datos aleatorios con errores de validación.

- Revisión de los modelos de riesgos y de la superficie de ataques: Se verifica si un producto no cuenta con las especificaciones funcionales y de diseño es de vital importancia que se realice una revisión de los modelos de riesgos.

- **Quinta etapa: Lanzamiento**

- Plan de respuesta a incidentes: se define los equipos de ingenieros con la

[71] FAN, R y CHANG,Y.Machine Learning for Black-Box Fuzzing of Network Protocols, En: Information and Communications Security, ed. Sihan Qing et al., Lecture Notes in Computer Science.9,No.19.2018.[En línea].Recuperado en 2020-621-32.Disponible en: https://doi.org/10.1007/978-3-319-89500-0_53.

capacidad de tomar decisiones e implementen estrategias para código a terceros.

- Revisión de seguridad final: no tiene en cuenta los siguientes procesos "penetrar y parchear"[72]. Consiste en realizar un análisis de los modelos de riesgos, revisión de excepciones, verificación de los resultados de las herramientas y el rendimiento teniendo en cuenta los estándares de calidad y los niveles de errores ya identificados.

- Lanzamiento o archivado: En esta etapa el asesor debe certificar que el proyecto que ha cumplido los requisitos ya establecidos de privacidad y los niveles de seguridad para que se pueda enviar el software.

5.1.2.3 Metodología opensamm

En la figura 4, la metodología OPENSAMM a través de sus funciones propone un modelo abierto donde las compañías deberán realizar un análisis de riesgos y amenazas, con esta metodología se podrá implementar una estrategia para resolver Los problemas de vulnerabilidad.

Figura 4.Modelo OPENSAMM

Fuente: GUASCH, José. Modelo de madurez de seguridad de software - opensamm en español, 2011. [Imagen digital].Gráfica digital.(5.72x17.65cm),Repositorio Institucional. Recuperado de:http://www.securitybydefault.com/2011/07/modelo-de-madurez-de-seguridad-de.htmll el 23 de febrero de 2020.

[72]RAMOS, B.Avances en criptología y seguridad de la información, En: Ediciones Diaz Santos.1,No.1.204.[En línea].Recuperado en 2020-15-59.Disponible en: https://books.google.com.co

El Modelo de Madurez de Software Assurance (OpenSAMM)[73], es un modelo abierto que permite a las organizaciones formular e implementar una estrategia para la seguridad del software. Este modelo intenta resolver, en específico los riesgos de seguridad de software que enfrenta la organización. La metodología OpenSAMM ayudarán a:

• Evaluar las prácticas de seguridad de software existentes de una organización.
• Crear un programa de garantía de seguridad de software equilibrado en iteraciones bien definidas
• Demostrar mejoras concretas a un programa de garantía de seguridad.
• Definir y medir las actividades relacionadas con la seguridad en toda la organización.

Cada función empresarial define tres prácticas de seguridad. Cada práctica de seguridad construye garantía para la función comercial relacionada. En general, hay doce prácticas de seguridad que son las independientes para la mejora que se asignan debajo de las funciones comerciales del software desarrollo En general, esto incluirá seguridad en cuanto requisitos, evaluación de amenazas y arquitectura segura[74].

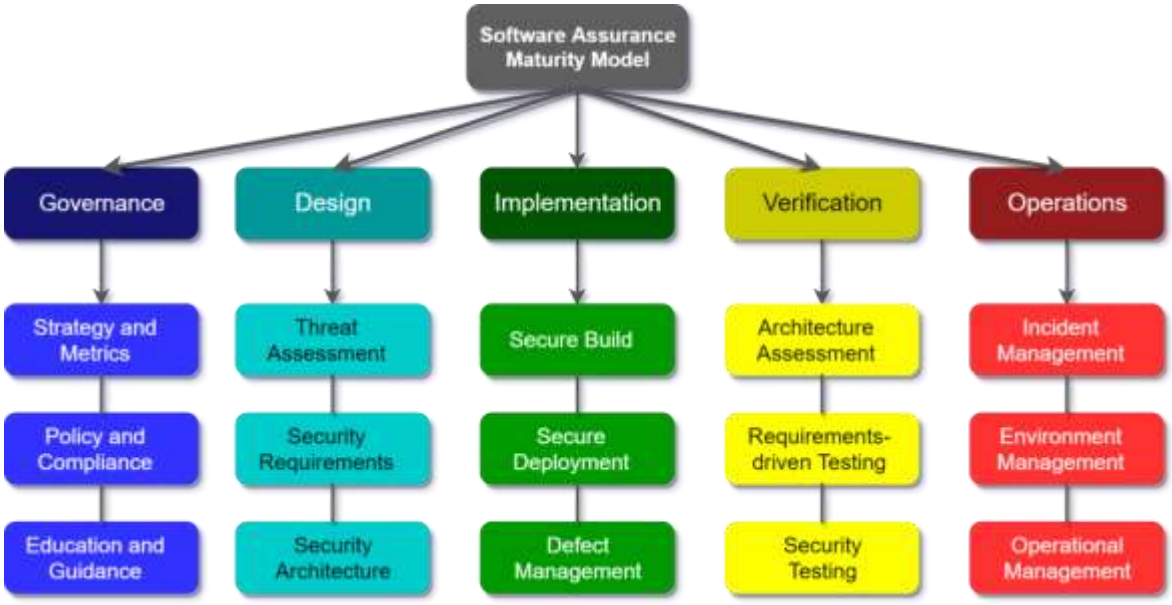
La verificación se refiere a los procesos y actividades relacionados con la forma en que una organización verifica y prueba de error producido en la fase de desarrollo de software. Esto se centra en la revisión del diseño, la seguridad. prueba y revisión de código. La implementación está interesada en los procesos y actividades relacionados con la gestión de una organización.

El lanzamiento del software que se ha creado. Esto puede implicar la entrega de productos a usuarios finales, al momento de desplegar productos en hosts internos o externos, y operaciones normales de software en el entorno de ejecución.
En la Figura4, se describe el modelo OpenSAMM que muestra un modelo creado como COBIT (Control Objetivo para la información y la tecnología relacionada)[75] que permite medir cuantitativamente cada seguridad objetivo. En este modelo, el nivel de madurez de la operación de seguridad toma un valor entre "0" y "3". "0".

[73] CHANDRA, P.Software Assurance Maturity Model (SAMM) a guide to building security into software development, En:OpenSamm Project Lead.1,No.1.2009.[En línea].Recuperado en 2020-15-38.Disponible en: http://www.cs.unh.edu/~it666/reading_list/SDLC/opensamm_1.0.pdf
[74] SIQUEIRA, P.A Security Testing Process Supported by an Ontology Environment: A Conceptual Proposal, En: International Conference on Computer Systems and Applications.15,No.1.2018.[En línea].Recuperado en 2020-1-8.Disponible en: https://doi.org/10.1109/AICCSA.2018.8612820.
[75] VON SOLMS, B. Information Security Governance: COBIT or ISO 17799 or Both, En:Computers & Security.24,No.2.2005.[En línea].Recuperado en 2020-99-104.Disponible en: https://doi.org/10.1016/j.cose.2005.02.002.

En la Figura 5, se describe el modelo OPEN SAMM en cuatro categorías elementales para la ejecución del proceso dentro de una organización.

Figura 5.Open SAMM model.



Fuente: MEHMET, Kara. Review on common criteria as a secure software development model,2015.[Imagen digital].Gráfica digital.(7.81x13.55cm),Revista de investigación ISSN.Recuperado de:https://www.researchgate.net/profile/Mehmet_Kara2/publication/267704821_Review_on_Common_Criteria_as_a_Secure_Software_Development_Model/links/55a5ef1008ae5e82ab1fca2a.pdf el 23 de febrero de 2020.

La metodología OPENSAMM, se define en cuatro categorías o actividades que conllevan un éxito al momento de ejecutar el proceso de desarrollo.

- Gobernabilidad: Se centra en los procesos políticos relacionados con las organizaciones

- Construcción: Se enfoca a procesos y actividades que tiene que ver con la administración del producto, recolección de requerimientos, especificaciones de la arquitectura a alto nivel.

- Verificación. Se centra en los procesos que se relaciona con la revisión y pruebas de las funcionalidades de los productos durante el desarrollo.

- Implementación. Se enfoca en las funcionalidades relacionadas con la liberación del producto.

I. Función de gobernabilidad

Según Burnstein, Suwanassarty Carlson[76], describe esta categoría como uno de los procesos y actividades relacionados con la gestión de una organización. actividades generales de desarrollo de software.

II. Función de construcción

Según Burnstein, Suwanassarty Carlson[77], menciona que esta categoría hace referencia a los procesos y actividades relacionados con la forma en que una organización define objetivos y crea software dentro de proyectos de desarrollo. En general, esto incluirá la gestión del producto.

III. Función de verificación

Según Burnstein, Suwanassarty Carlson[78], se centra en los procesos y actividades relacionadas con la forma en que una organización verifica y prueba artefactos producidos a lo largo del desarrollo de software. Esto generalmente incluye el trabajo de aseguramiento de la calidad, como las pruebas, pero también puede incluir otras actividades de revisión y evaluación

IV. Función de implementación

Implica los procesos y actividades relacionadas con la forma en que una organización gestiona la liberación del software que se ha creado. Esto puede implicar el envío de productos a los usuarios finales, la implementación de productos en hosts internos o externos y las operaciones normales de software en el entorno de tiempo de ejecución[79].

[76]BURNSTEIN, I.;SUWANNASSART,T y CARLSON ,R.Developing a Testing Maturity Model for software test process evaluation and improvement, En:Proceedings International Test Conference.1,No.3.1996.[En línea].Recuperado en 2020-89-581.Disponible en: https://doi.org/10.1109/TEST.1996.557106.
[77]Ibid.,p.42
[78]Ibid.,p.42
[79] MARK,P.Capability Maturity Model for Software,En: Encyclopedia of Software Engineering (American Cancer Society).15,No.1.2002.[En línea].Recuperado en 2020-392-401.Disponible en: https://doi.org/10.1016/j.infsof.2008.01.010.

5.1.2.4 Metodología building security in maturity model (BSIMM2)

Desde el lanzamiento del Modelo original de seguridad de la construcción en madurez (BSIMM) en 2009, el tamaño del estudio se ha triplicado. BSIMM2 es la segunda iteración del proyecto BSIMM y se puede utilizar como un dispositivo de medición para la seguridad del software.

McGraw, Chess, Migues y Nichols[80], lo describen como un modelo de seguridad maduro, debido que BSIMM2[81], se puede utilizar como un dispositivo de medición para la seguridad del software. Como tal, es útil para comparar las actividades de seguridad de software con el objetivo de, realizar una comparación directa usando el BSIMM como una excelente herramienta para diseñar una estrategia de seguridad de software.

BSIMM es una forma en que las organizaciones pueden encontrar un "cronómetro" para la evaluación de la seguridad del software. Es una herramienta de medición que las organizaciones y las empresas pueden usar para comprender cómo un, software se compara con los niveles de seguridad relativos de software escritos internamente, por otras organizaciones y contra todo el portafolio de aplicaciones de software que las organizaciones están implementando. que han elegido participar en el proyecto BSIMM[82].

Con la gran cantidad de metodologías de desarrollo de software disponibles, no es sorprendente que algunas personas confundan BSIMM con otra metodología más entre las muchas. Pero BSIMM no es una metodología de ciclo de vida de desarrollo de software como Touchpoints de Cigital[83], SDL de Microsoft, IBM Rational[84] o el proceso de seguridad de aplicaciones ligero y completo (CLASP) de OWASP. BSIMM es un modelo descriptivo que observa los datos asociados con el desarrollo

[80] SOBEL,A y MCGRAW,G.Interview: Software Security in the Real World, En: Computer.43,No.9.2010.[En línea].Recuperado en 2020-47-53.Disponible en: https://doi.org/10.1109/MC.2010.256.

[81] BERNSMED, K .;GILIE,M y HAKON,P.Safety Critical Software and Security - How Low Can You Go?, En: IEEE/AIAA 37th Digital Avionics Systems Conference.37,No.1.2018.[En línea].Recuperado en 2020-1-6.Disponible en: https://doi.org/10.1109/DASC.2018.8569579.

[82] KELLEY, D.Measuring Software Security: BSIMM2 and Beyond,eSecurty, En: eSecurty Planet .37,No.1.2010.[En línea].Recuperado en 2020-1-6.Disponible en: https://www.esecurityplanet.com/views/article.php/3881771/Measuring-Software-Security-BSIMM2-and-Beyond.htm.

[83] STRAKER, K .;WRIGLEY,C y ROSEMANN, M.Typologies and Touchpoints: Designing Multi-Channel Digital Strategies, En:Journal of Research in Interactive Marketing.9,No.2.2015.[En línea].Recuperado en 2020-110-28.Disponible en: https://doi.org/10.1108/JRIM-06-2014-0039.

[84] BROWN, A.Realizing service-oriented solutions with the IBM Rational Software Development Platform, En:IBM.44,No.4.2005.[En línea].Recuperado en 2020-727-52.Disponible en: https://doi.org/10.1147/sj.444.0727.

y la seguridad del software, y registra esa información de la manera más objetiva posible.

El proyecto BSIMM2 ha agregado datos de medición de seguridad de software de más de 30 iniciativas de seguridad de software con más de diez más en curso. Las más de 30 organizaciones.

Han dividido los puntos de medición de 109 actividades en 12 prácticas principales:

1. Estrategia y métrica
2. Cumplimiento y política
3. Entrenamiento
4. Modelos de ataque
5. Características de seguridad y diseños
6. Normas y requisitos
7. Análisis de arquitectura
8. Revisión de código
9. Pruebas de seguridad
10. Pruebas de penetración
11. Entorno de software
12. Gestión de vulnerabilidades y gestión del cambio

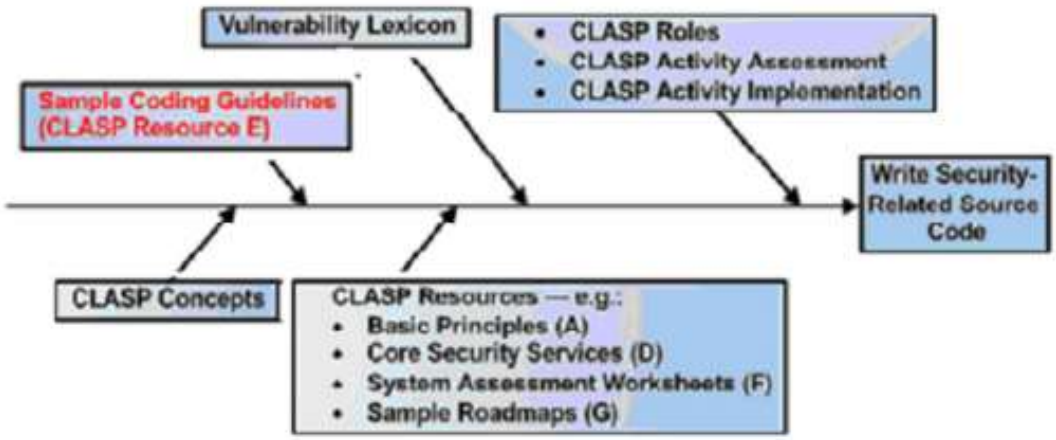
5.1.2.5 Metodología comprehensive lightweight application security process (CLASP)

Modelo prescriptivo[85], esta metodología se basa en perfiles y en conjunto de buenos lineamientos, debido que los ingenieros de desarrollo podrán implementar la seguridad en cada fase de manera estructurada, medible, adicional se constituye como un proceso ligero donde se incluye 24 actividades de alto nivel.

[85] MERA, C.Concepto, aplicación y modelo de prospectiva estratégica en la administración de las organizaciones, En:Revista Estrategia Organizacional.1,No.4.2012.[En línea].Recuperado en 2020-25.Disponible en: https://doi.org/10.22490/25392786.1208.

Las actividades de la metodología CLASP están descritas en la siguiente Figura 6, debido que está definido en un modelo basado en perfiles o roles dentro de un grupo de buenos lineamientos.

Figura 6.Actividades CLASP



Fuente: ROBLES, Ericka.Ciclo de Vida de Desarrollo de Software Seguro en Metodologías Ágiles ,2011.[Imagen digital].Gráfica digital.(6.56x14.37cm),México, CIMAT, Repositorio Institucional. Recuperado de:https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/411/1/ZACTE14.pdf el 26 de febrero de 2020.

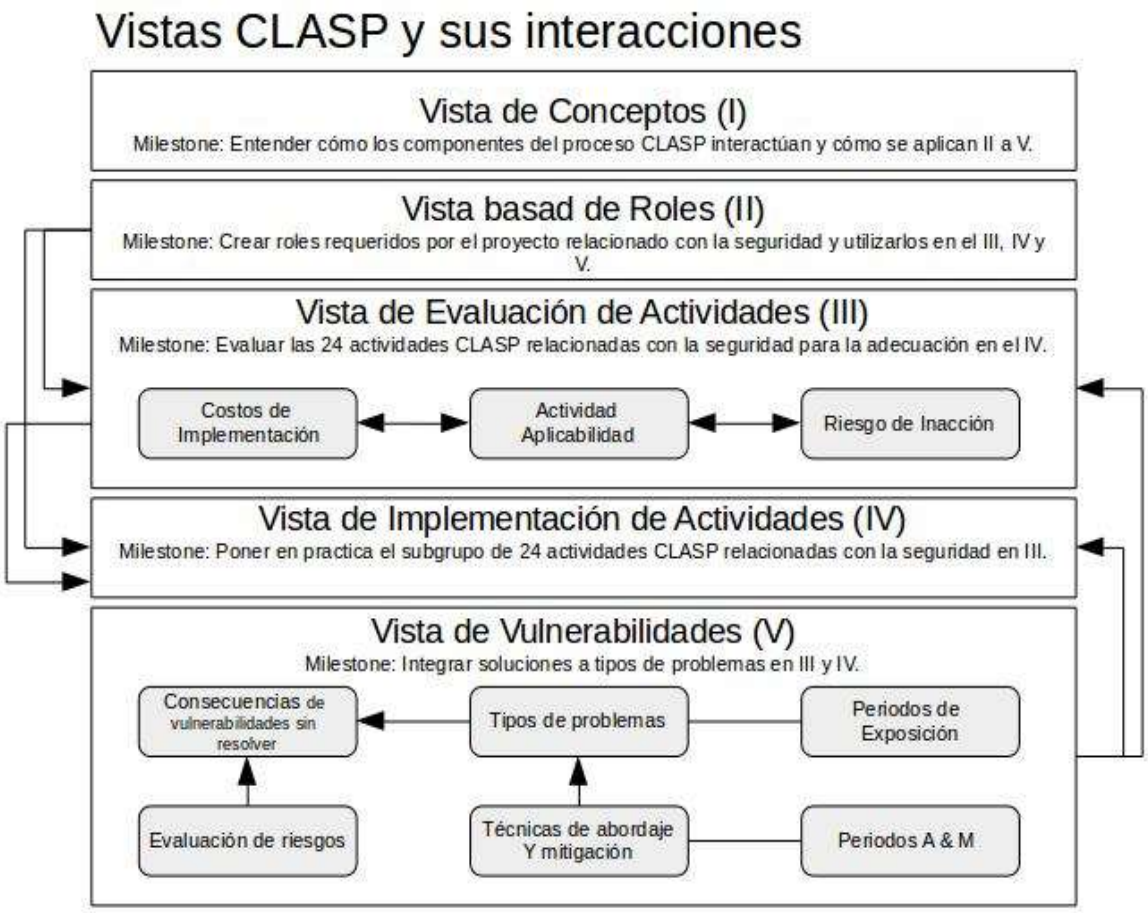
Estructura metodología CLASP

A continuación, se relaciona estructura que hace parte de la metodología CLASP a través de:

- Vistas del proceso.
- Agrupación de actividades.
- Perspectivas de alto nivel
- 24 actividades
- Recursos
- Taxonomy.

Vistas del proceso CLASP: Esta estructura está dividida en cinco perspectivas de alto nivel llamadas vistas que se encuentran desglosadas por actividades, donde cada vista la compone elementos que facilitan el proceso de manera ágil. Como se muestra en la siguiente Figura 7:

Figura 7.Explicación proceso CLASP



Fuente: ROBLES, Ericka. Explicación proceso CLASP,2011. [Imagen digital]. Gráfica digital. (15.58x15.82cm),Mexico, CIMAT, Repositorio Institucional. Recuperado de:https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/411/1/ZACTE14.pdf. el 26 de febrero de 2020

Vista conceptos: Se describe como un conjunto de componentes de procesos que se basa en perfiles enfocados a actividades. En cuyo centro contiene, las mejores prácticas para materializar la seguridad en cada uno de los ciclos vida de desarrollo de software. Esto de una manera estructurada, repetible y medible. La vista de conceptos se divide en:

- Descripción del proceso CLASP: Cuenta con una descripción estructurada que depende de cada uno de sus componentes.
- Buenas prácticas: materializa una alternativa que sea razonable para el uso de las mejores prácticas en cuanto en seguridad y que se posible incluir dentro del ciclo de vida del desarrollo.
- Descripción taxonómica: se define como categorización de alto nivel con el objetivo de mejorar la evaluación y resolución de problemas de seguridad.
- Vista Basada en roles: Esta vista se enfoca, en una visión especializados en cada uno de los roles que hacen parte de los equipos de ingeniera, como por ejemplo; administrador de proyectos, diseñador, especificador de requerimientos, analista de pruebas etc.

- Vista de evaluación de actividades. El objetivo de esta actividad es, monitorear las actividades que realiza el administrador de proyectos y el equipo de ingeniería de desarrollo con el fin de proporcionar una guía que ayude a evaluar las 24 actividades propuestas por la metodología CLASP.

- Vista de implementación de actividades. Esta vista corresponde al centro de la metodología CLASP, debido que se requiere implementar las 24 actividades e incluirlas dentro del ciclo de vida del desarrollo.

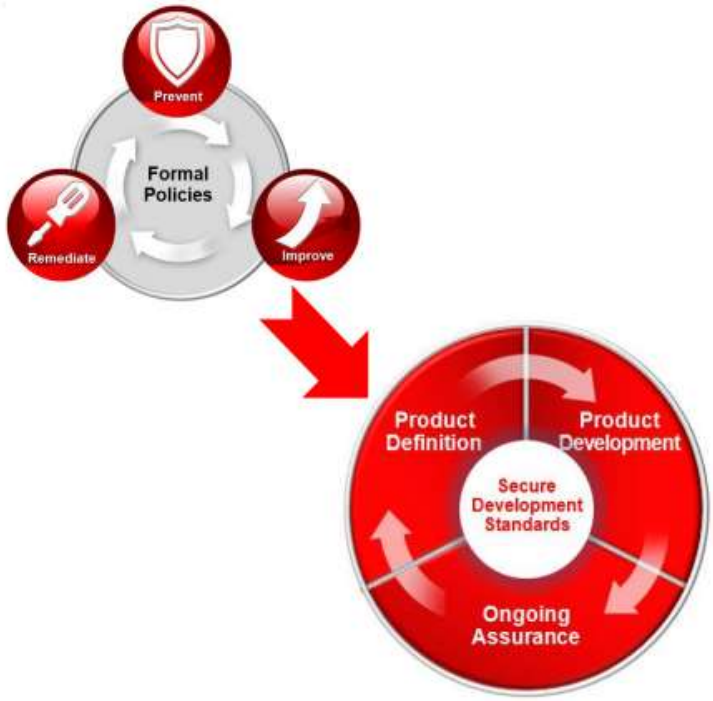
- Vista de vulnerabilidades: En esta vista se prioriza los temas que se centran en los problemas de seguridad mediante, una categorización de alto nivel o taxonomía, incluyendo los siguientes criterios sobre cada vulnerabilidad; fase de mitigación y liberación, categoría, tipo de problema, consecuencias, plataformas, evaluación de requisitos.

5.1.2.6 Metodología oracle software security assurance (OSSA)

Es la metodología de Oracle[86], esta metodología permite garantizar que los productos cumplan con cada uno de los requisitos que se enfocan en seguridad, debido que proporcionan una experiencia de costo más efectivo.

En la Figura 8, se describe como está estructurado esta metodología donde el enfoque es el producto desarrollado, esta metodología brinda tranquilad ya que Oracle se encuentra en constante evolución en cuanto sus procesos, procedimientos y tecnologías que garantiza productos de alta calidad.

Figura 8.Estructura metodología OSSA



Fuente:
HEIMANN, John. Estructura metodología OSSA,2014. [Imagen digital]. Gráfica digital. (11.59x12.75cm),California, ORACLE, Repositorio Institucional. Recuperado de: https://conference.usu.edu/SYSTEM/Uploads/pdfs/14612_2106JohnHeimann.pdf el 26 de febrero de 2020.

[86] DONG, G.Concepto, Security Assurance with Metamorphic Testing and Genetic Algorithm, En:IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.1,No.4.2010.[En línea].Recuperado en 2020-397-401.Disponible en: https://doi.org/10.1109/WI-IAT.2010.101.

A medida que las compañías se sientan más confiadas en cuanto la implementación los controles más rigurosos para proteger sus entornos de desarrollo y datos, asegurando que estos controles operen en la forma en que fueron pensadas se ha convertido en una preocupación primordial. La garantía de seguridad abarca todas las actividades involucradas en asegurar que el software opera a un nivel de seguridad esperado y predecible.

Gestión de incidentes de seguridad, cumplimiento de regulaciones y auditorías complejas, requisitos y mantenerse al día con las versiones de seguridad pueden agregar enormes costos para el presupuesto de TI de una organización, por lo tanto, las organizaciones deben considerar cuidadosamente cómo sus proveedores de software se acercan al software garantía de seguridad[87].

Oracle Software abarca todas las fases del ciclo de vida del desarrollo de productos. Security Assurance es la metodología de Oracle para incorporar seguridad en el diseño, construcción, prueba, entrega y mantenimiento de sus productos. El objetivo de Oracle es asegurar que los productos de Oracle, así como los sistemas de los clientes que aprovechan esos productos, permanecer lo más seguro posible.

Objetivos (OSSA)

•Innovar en cuanto seguridad: Esta metodología fue diseñada por Oracle, gracias a la marca se convierte en tradición larga de innovación, ayudando a las compañías a implementar y administrar políticas de seguridad.

•Mitigar la incidencia de las deficiencias que se produce en todos los productos**:** Se incluyen una serie de estándares de codificación que sea de manera segura, con el uso de herramientas automatizadas de análisis de Oracle.

•Mitigar el impacto incidencias hacia el usuario final: Oracle adopto una seria de políticas que se enfocan en la corrección de las vulnerabilidades presentadas en los productos de Oracle, tratando a todos los clientes por igual ofreciendo actualizaciones.

A continuación, se describe unos estándares de codificación segura

Oracle da a conocer una guía de buenas prácticas que los desarrolladores deberían tener en cuenta al momento de producir un código con estándares de seguridad, enfocados en los siguientes criterios de diseño en cuanto principios y

[87] ZEESHAN, S .;ADDUL,H y MUHAMMAD,K.ualified Analysis b/w ESB(s) Using Analytical Hierarchy Process (AHP) Method, En:Second International Conference on Intelligent Systems, Modelling and Simulation.2,No.1.2011.[En línea].Recuperado en 2020-100-104.Disponible en:https://doi.org/10.1109/ISMS.2011.25

vulnerabilidades comunes, con el objeto de proporcionar una orientación en cuanto la validación de datos, privacidad de información, administración de accesos[88].
Los estándares de codificación segura se definen como un componente clave de Oracle Software Security Asúrense donde se evalúa el que se cumpla los criterios ya definidos y realizando una validación en toda la vida útil de todos los productos de Oracle.

Análisis y pruebas integrales de seguridad (OSSA)

Las pruebas de seguridad en Oracle incluyen actividades tanto funcionales como no funcionales para la verificación de las características y la calidad del producto.

Aunque este tipo de pruebas a menudo apuntan características de productos superpuestas, tienen objetivos diferentes, por lo tanto, se llevan a cabo por diferentes equipos. Las pruebas de seguridad funcionales y no funcionales se complementan entre sí para proporcionar cobertura de seguridad integral de Oracle.

Las pruebas de seguridad funcional generalmente son ejecutadas por equipos regulares de control de calidad, del producto como parte de ciclo normal de prueba del producto. Durante esta prueba, los ingenieros de control de calidad verifican las especificaciones funcionales, durante este proceso de revisión arquitectónica y de la lista de verificación.

El análisis y las pruebas de seguridad no funcionales[89], verifican la garantía de seguridad de los productos, incluida la identificación y reparación de vulnerabilidades y resistencias a los ataques, hay dos amplias categorías de pruebas empleadas, para probar los productos de Oracle: análisis estático y dinámico. Estas pruebas se ajustan de manera diferente al ciclo de vida de desarrollo y tienden a encontrar diferentes categorías de problemas, por lo que se utilizan equipos de seguridad de Oracle.

[88]KIM, H .;WEN,J y VILLASENOR,J.Secure Arithmetic Coding, En:IEEE Transactions on Signal Processing.55,No.5.2007.[En línea].Recuperado en 2020-2263-72.Disponible en:https://doi.org/10.1109/TSP.2007.892710.
[89] NGUYEN, Q.Concepto, Non-functional requirements analysis modeling for software product lines, En:ICSE Workshop on Modeling in Software Engineering.1,No.4.2009.[En línea].Recuperado en 2020-56-61.Disponible en: https://doi.org/10.1109/MISE.2009.5069898.

5.1.2.7 Metodología team software process (TSP)

El Team Software Process (TSP), está metodología, se puede decir que es un proceso de desarrollo enfocado, especialmente para equipos de ingenieros debido que se basa en la modelo internacional en CMMI[90]. TSP proporciona directrices para resolver problemas que ayudan a los equipos de ingeniera establecer prácticas de ingeniera avanzada, adicional TSP presta una solución debido que optimiza los tiempos de desarrollo, mejora la calidad de los productos.

En la Figura 9, se describe el modelo organizacional que deben tener cuenta las compañías si desean implementar esta metodología teniendo en cuenta los elementos de diciplina, calidad, tiempo, costo y competencias dado este modelo permite tener un mayor control dentro de sus procesos internos.

Figura 9.Model Team Software Process Secure

[90] HSUEH, N. Concepto, Applying UML and Software Simulation for Process Definition, Verification, and Validation, En: Information and Software Technology.50, No.9.2008.[En línea].Recuperado en 2020-897-911.Disponible en: https://doi.org/10.1016/j.infsof.2007.10.015

Fuente: NOOPUR,Davis.Model Team Software Process Secure,2005.[Imagen digital].Gráfica digital.(7.57x10.93cm), Carnegiemelloii software engineering institute, Repositorio Institucional..Recuperado de:Disponible en: https://apps.dtic.mil/dtic/tr/fulltext/u2/a447047.pdf el 27 de febrero de 2020.

Sus orígenes se deben a las limitaciones que el PSP [91](Personal Software Process, su antecesor), tenía en el ámbito industrial. PSP resultó muy efectivo para que los ingenieros, ya que podían tener el control de su proceso personal mediante la mejora de sus habilidades de estimación y la reducción de los defectos introducidos en los productos sin afectar a su productividad, pero PSP sólo se enfocaba en las fases de desarrollo de software (diseño y pruebas unitarias).

En la Figura 10, presenta una comparación entre los elementos fundamentales dentro de la metodología como es TSP y PSP dado que cada uno cuenta con características fundamentales para la ejecución de esta metodología dentro de los equipos de trabajo.

Figura 10.Comparación TSP y PSP

Fuente: NOOPUR,Davis. Comparación TSP y PS,2005. [Imagen digital].Gráfica digital.(7.57x13.23cm), CARNEGIEMELLOII SOFTWARE ENGINEERING INSTITUTE, Repositorio Institucional. Recuperado de: Disponible en: https://apps.dtic.mil/dtic/tr/fulltext/u2/a447047.pdf el 27 de febrero de 2020.

[91] HUMPHEREY, W.Team Software Process (TSP), En:Encyclopedia of Software Engineering American Cancer Society.1,No.1.2002.[En línea].Recuperado en 2020-897-911.Disponible en:https://doi.org/10.1002/0471028959.sof352.

TSP para el desarrollo de software seguro (TSP-Secure)[92], extiende el TSP para centrarse más directamente en la seguridad de las aplicaciones de software. El proyecto TSP-Secure es un esfuerzo conjunto de la iniciativa TSP de SEI y el programa CERT de SEI.

Dentro del desarrollo seguro TSP tiene presente los siguientes criterios:

- Primero que todo un software seguro nunca se crea por accidente, debido TSP-Secure se basa a través de una planificación de seguridad.
- En segundo lugar, TSP se enfoca en desarrollar un producto que cumpla con todos los niveles de calidad en cuanto seguridad.
- Finalmente, Los equipos de desarrollo deben contar con los conocimientos con un nivel alto, al momento de solucionar problemas

Dentro de las fases TSP se divide en una serie de ciclos que se mencionan a co.ntinuación:

1. Lanzamiento
2. Estrategia
3. Plan
4. Requisitos
5. Diseño
6. Implementación
7. Pruebas
8. Postmortem

5.1.2.8 Metodología correctness by construction (CBYC)

Según el autor Hilbrich[93], expone que los sistemas de software de alta integridad a menudo son tan grandes que los procesos de desarrollo convencionales no pueden llegar a alcanzar tasas de defectos tolerables. Se presenta la corrección por construcción (CbyC), un enfoque que ha proporcionado un software de tasa de defectos muy bajo de manera rentable. Describimos los principios fundamentales de CbyC y los resultados logrados hasta la fecha. También tocamos algunas de las barreras que hemos encontrado al tratar de colocar CbyC dentro de nuestras propias organizaciones y otras.

---

[92] DAVIS,N.Processes for producing secure software, En:IEEE Security & Privacy Magazine.2,No.3.2004.[En línea].Recuperado en 2020-897-911.Disponible en:https://doi.org/10.1109/MSP.2004.21.

[93] HILBRICH,R.How to Safely Integrate Multiple Applications on Embedded Many-Core Systems by Applying the Correctness by Construction Principle,En:Advances in Software Engineering.2,No.3.2012.[En línea].Recuperado en 2020-1-14.Disponible en:https://doi.org/10.1155/2012/354274.

CbyC se basa en tres principios simples:

- Haz que sea muy difícil introducir errores.
- Asegúrese de que los errores, se eliminen lo más cerca posible del punto de introducción, ya que se realizarán a pesar del elemento 1.
- Genere evidencia de aptitud para, el propósito a lo largo del desarrollo como un subproducto natural del proceso, (porque demostrar que el sistema desarrollado es seguro o seguro a menudo es más difícil que hacerlo).

CbyC, es un método técnico para el desarrollo de software, que es altamente compatible con los principios de proceso de PSP / TSP. La evidencia provisional a pequeña escala muestra que CbyC combinado con PSP / TSP puede dar como resultado tasas de defectos aún más bajas. El enfoque técnico de CbyC puede complementar PSP para proporcionar altos rendimientos de proceso.

Los pasos del proceso

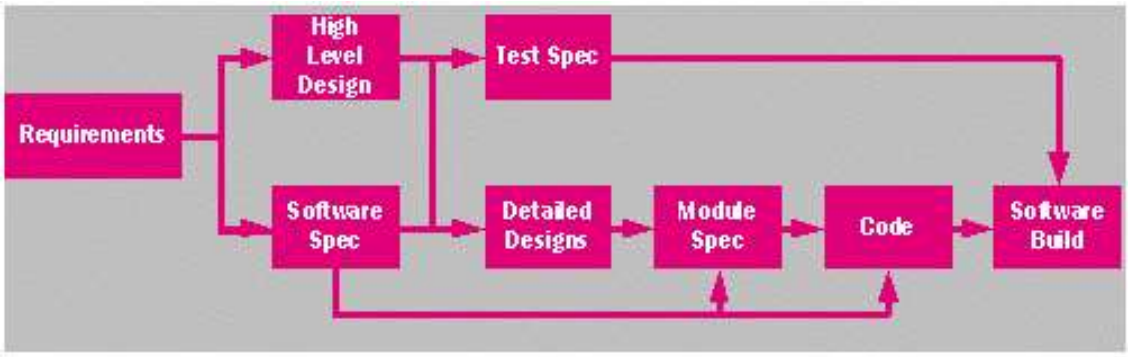
Entonces, ¿cómo organizamos los bloques de construcción anteriores en un proceso de desarrollo coherente? La Figura14 es un diagrama simplificado del proceso central de CbyC.

- Eliminación de gran parte del paralelismo y superposición entre las etapas del proceso.

- Omisión de comentarios de las diversas actividades de validación que pueden afectar claramente cualquier entrega anterior y causar el reingreso a cualquier actividad previa.

En la siguiente Figura 11, muestra los requerimientos necesarios que se requiere tener cuenta si las organizaciones implementen esta metodología dado que es compatible con los principios de proceso de PSP / TSP.

Figura 11.Core Correctness by Construction Process



Fuente: AMEY, Petter. Core Correctness by Construction Process,2006. [Imagen digital]. Gráfica digital.(4.23x13.28cm), Raxis high integrity systems, Repositorio Institucional. Recuperado de: https://www.us-cert.gov/bsi/articles/knowledge/sdlc-process/correctness-by-construction el 27 de febrero de 2020.

Elementos clave (CbyC)

- Los requisitos: Los datos relevantes sobre el dominio de la aplicación también deben ser capturados.
- El diseño de alto nivel: Se abordan las propiedades no funcionales clave, como la seguridad y la protección. También es el punto en el que proporcionamos incertidumbres de requisitos, al seleccionar un diseño que sea flexible en áreas de probable cambio. De manera poco intuitiva, la respuesta de CbyC a la incertidumbre de los requisitos es más diseño, no menos.
- La especificación de prueba: se realiza la especificación de software, en conjunto con los requisitos y el diseño de alto nivel. El análisis del valor límite se utiliza para generar pruebas que cubren la especificación, que se complementan con pruebas de comportamiento introducidas por el diseño pero que no son visibles en la especificación. Además, las pruebas de requisitos no funcionales se generan directamente, desde el documento de requisitos. Debido a la fuerte validación y a las pequeñas brechas semánticas, inherentes al enfoque CbyC, no es útil realizar pruebas unitarias, basadas en códigos o en cajas blancas; todas las pruebas en CbyC se basan en el nivel del sistema y las especificaciones.

- El software instalado: se define como la forma de integración continua y validación, debido que evita problemas de última hora (es decir, cuello de botella de integración).

5.1.3 Identificación de características

Se procede a listar las principales características con las que cuentan las metodologías para el desarrollo de software seguro.

5.1.3.1 Características de las metodologías (SDL)

Los pasos básicos del modelo SDL se muestran en la Figura3. Microsoft ha agregado muchas propiedades nuevas y capacidad desde 2002.
Importante de esta capacidad, como barra de errores, fuzzing, criptográficos estándares, pruebas de verificación de tiempo de ejecución, API prohibida (interfaz del programa de aplicación), privacidad estándar para desarrollo, requisitos de servicio en línea, defensas de scripts de sitios cruzados, SQL defensas de inyección, defensas de análisis XML, aleatorización del diseño del espacio de direcciones, sitio cruzado[94].

A continuación, se menciona las principales características que hacen parte de esta metodología:

- Formación: Se destaca la necesidad de divulgar de forma acertada la filosofía de la seguridad a todos los roles que intervienen en el proceso que cuenten los conocimientos técnicos que sean necesarios.
- Diseño seguro: Esta característica permite reducir la margen de ataques fortalecer la defensa en profundidad.
- Modelos de riesgos: En esta característica permite saber la Información general sobre los modelos de riesgos, las implicaciones de que se presente en la etapa de diseño de un modelo de riesgos.
- Codificación segura: Esta característica se enfoca en indicar una serie de buenas prácticas que se debe incluir dentro del código.
- Pruebas de seguridad: En esta característica se establece una diferencia entre las pruebas de seguridad y pruebas funcionales.

[94] GEPPERT,B y ROBLER,F.The SDL Pattern Approach – a Reuse-Driven SDL Design Methodology,En: Computer Networks .35,No.6.2001.[En línea].Recuperado en 2020-45-627.Disponible en: https://doi.org/10.1016/S1389-1286(00)00202-4.

- Privacidad: Se define los tipos de datos que sean confidenciales, se elaboran procedimientos de diseño de enfocados en la privacidad[95].

5.1.3.2 Características opensamm

Este modelo se enfoca en desarrollar un software de acuerdo con el negocio, cuenta con 3 niveles de madurez, definidos a través de 12 de prácticas de seguridad que se deben ser implementadas en las compañías para reducir riesgos e incrementar el aseguramiento del software[96].

Los recursos proporcionados por OPENSAMM ayudarán a:

- Evaluar la existencia de las prácticas de seguridad de software que deben estar implementadas en la gran parte de las compañías.
- Crear un programa de seguridad de software equilibrado en iteraciones bien definidas.
- Demostrar mejoras concretas a un programa de garantía de seguridad.
- Definir y medir actividades relacionadas con la seguridad dentro de una organización.

Este modelo SAMM fue diseñado para ser una manera flexible, de tal manera que sea sencillo adaptarlo en cualquier tamaño de compañía y fue diseñado bajo los siguientes criterios:

- Cambios espaciosos en la organización: Este criterio permite que se cree un programa de seguridad que sea exitoso e implementado en pequeñas iteraciones, lo cual permitirá crear entregables tangibles y de valor para la organización en un tiempo más reducido.
- No existe un producto único que funcione en todas las compañías, este marco de trabajo se enfoca en seguridad de software, este debe ser forma flexible y adicional se pueda incluir controles necesarios basándose en el nivel de riesgo de la organización.
- Establecimiento de una directriz de seguridad: Se establece una serie de actividades de un programa que permita realizar aseguramientos que deben estar bien definidos.

---

[95]DONGGILL,L.A New Integrated Software Development Environment Based on SDL, MSC, and CHILL for Large-Scale Switching Systems,En:ETRI Journal.18,No.4.1997.[En línea].Recuperado en 2020-86-265.Disponible en: https://doi.org/10.4218/etrij.97.0197.0044.

[96] CARRIZO,D y ALFARO,A.Método de Aseguramiento de La Calidad En Una Metodología de Desarrollo de Software: Un Enfoque Práctico,En:Revista Chilena de Ingeniería.26,No.1.2018.[En línea].Recuperado en 2020-29-114.Disponible en: https://doi.org/10.4067/S0718-33052018000100114.

5.1.3.3 Características building security in maturity model (BSIMM2)

Al revisar los niveles de madurez, dentro de las organizaciones se puede determinar qué, tan equilibrado está enfocado en relación con los demás. Por ejemplo, en el área de Revisión de Código (CR) hay 3 niveles de madurez que se parafrasean a continuación[97]:

1. Hacer la revisión del código
2. Aplicar los estándares a través de la revisión obligatoria de código automatizada
3. Revisión automática de código con reglas personalizadas (a medida).
4. Hay un concepto claro sobre lo que es “lo que hay que hacer”, donde llama la atención de todos los involucrados del proceso-
5. Se minimiza los costos mediante procesos estándar y repetible.
6. Sobresale la calidad de un código bien hecho.

En la Figura 12, se enuncian las diferentes prácticas que se ejecutaron antes de definir que la metodología BSIMM2, cumpliera con los niveles de madurez que se requería para ser una metodología de desarrollo seguro.

[97] ALMEIDA, J.;NETO,J y QUEIROS,L.A construção de software seguro: uma análise a partir dos modelos de desenvolvimento e maturidade, En:Revista Eletrônica da Estácio Recife.2,No.2.2016.[En línea].Recuperado en 2020-89-581.Disponible en: https://reer.emnuvens.com.br/reer/article/view/84.

Figura 12.Practicas BSIMM2

domain	practice	business goals
Governance	Strategy and Metrics	Transparency of expectations, Accountability for results
	Compliance and Policy	Prescriptive guidance for all stakeholders, Auditability
	Training	Knowledgeable workforce, Error correction
Intelligence	Attack Models	Customized knowledge
	Security Features and Designs	Reusable designs, Prescriptive guidance for all stakeholders
	Standards and Requirements	Prescriptive guidance for all stakeholders
SSDL Touchpoints	Architecture Analysis	Quality control
	Code Review	Quality control
	Security Testing	Quality control
Deployment	Penetration Testing	Quality control
	Software Environment	Change management
	Vulnerability Mgmt and Change Management	Change management

Fuente MCGRAW, Gary.BSIMM2 Offers Key Elements of Successful Software Security Initiatives Shared by 30 Major Corporations,2010. [Imagen digital].Gráfica digital.(4.23x13.28cm),Carnegie Mellon University CyLab. Recuperado de: http://www.cyblog.cylab.cmu.edu/2010/05/evolving-rapidly-bsimm2-offers-key.html el 28 de febrero de 2020.

5.1.3.4 Características Comprehensive Lightweight Application Security Process (CLASP)

Esta metodología, se define como un proceso ligero al momento de diseñar un software con seguridad gracias al conjunto de actividades que cuenta de alto nivel, a continuación, se describe sus principales características[98].

- El objetivo principal de CLASP es apoyar en la construcción de software, en donde la seguridad tiene un papel central. Las actividades de CLASP se definen y conciben principalmente desde la perspectiva de la seguridad teórica y, por tanto, la cobertura del conjunto de actividades es bastante amplia.
- CLASP se define como un conjunto de actividades independientes, que tienen que ser integradas en el proceso de desarrollo y de su entorno de trabajo. Dos rutas (“legado” y “Green-field”) se han definido para dar una orientación sobre cómo combinar las actividades en un conjunto coherente y ordenado.

[98] ROBLES,E.Ciclo de Vida de Desarrollo de Software Seguro en Metodologías Ágiles ,En: Repositorio CIMAT.1,No.1.2011.[En línea].Recuperado en 2020-18-21.Disponible en: http://cimat.repositorioinstitucional.mx/jspui/handle/1008/411.

- CLASP, proporciona un amplio, conjunto de recursos de seguridad que facilitan y apoyan la ejecución de las actividades. Uno de estos recursos es una lista de 104 vulnerabilidades de seguridad conocidas en el código fuente de aplicación.

5.1.3.5 Características oracle software security assurance (OSSA)

El programa Critical Patch Update[99], es el mecanismo principal para el lanzamiento de correcciones de errores de seguridad para productos Oracle. Las actualizaciones críticas de parches se lanzan trimestralmente, además Oracle conserva la capacidad de emitir parches o soluciones fuera de horario instrucciones en caso de vulnerabilidades particularmente críticas. Este programa se conoce como el programa de alerta de seguridad. Los procedimientos en cuanto la gestión y corrección de vulnerabilidades de seguridad de Oracle están destinados a brindar a los clientes los siguientes beneficios:

- Máxima seguridad: Oracle remedia las vulnerabilidades en orden de gravedad, este proceso garantiza que los agujeros de seguridad más críticos se reparen primero en la actualización crítica de parches, optimizando así la postura de seguridad de todos los Oracle clientes.
- Costos administrativos más bajos: un cronograma fijo de parches de seguridad al momento de la gestión de parches.
- Administración de parches simplificada: las actualizaciones de parches son acumulativas para la mayoría de Oracle productos Esto proporciona a los clientes la capacidad de "ponerse al día", rápidamente con el actual nivel de versión de seguridad, desde la aplicación del último parche crítico acumulativa, la actualización resuelve todas las vulnerabilidades tratadas anteriormente.
- Mantener la postura de seguridad de TODOS los clientes de Oracle son unas de las mayores prioridades de Oracle.
- Se aplica a TODO el software de Oracle productos a lo largo de su ciclo de vida, incluyendo componentes de software de productos de hardware (por ejemplo, firmware) y soluciones en la nube.
- Evoluciona constantemente para adaptarse a nuevos tecnologías, amenazas y producto casos de uso.

[99]KOST,S.Oracle Critical Patch Updates: Insight and Understanding,En:Integrigy Corporation.25,No.1.2011.[En línea].Recuperado en 2020-1-32.Disponible en: https://www.integrigy.com/files/Integrigy_Oracle_CPU_October_2011_Oracle_Database_Impact%20v1.pdf

5.1.3.6 Características team software process (TSP)

La implementación de la metodología TSP, dentro de las compañías ha sido beneficiadas dentro de sus procesos de calidad, debido que es muy útil para el desarrollo de proyectos de ingeniería enfocados en la seguridad por consiguiente, se menciona las principales características que conllevan el éxito al momento de aplicar esta esta metodología[100].

- Diseñar y generar un nuevo marco basado en PSP.
- Crear software seguro en varios ciclos aplicando las directrices dadas por el modelo.
- Se diseña estándares para generar métricas analíticas en cuanto al comportamiento y en calidad.
- Generar evaluaciones periódicas de desempeño a cada uno de los roles y equipos.
- Generar y aplicar guías con solución a problemas ya previamente identificados.

5.1.3.7 Características correctness by construction (CBYC)

Los objetivos principales de esta metodología es obtener una taza de defectos al mínimo, donde sea difícil introducir errores y que estos sean removidos tan pronto se identifiquen con el apoyo de requerimientos rigurosos de seguridad, a continuación, se menciona las principales características[101]:

Validación fuerte

- Debido que CbyC, utiliza notaciones precisas e inequívocas, es posible utilizar métodos sólidos y compatibles con herramientas para verificar, los picos de aquellos resultados que se generaron al ejecutar cada fase, por

[100] CAMACHO,J.Prototipo Web para la Gestión de Métricas como Soporte a la Metodología de Desarrollo TSP ,En: Repositorio Institucional Universidad Distrital - RIUD .6,No.1.2017.[En línea].Recuperado en 2020-309-401.Disponible en: http://repository.udistrital.edu.co/handle/11349/6236.

[101] BRITO, C.Metodologías para desarrollar software seguro, En: Universidad Autónoma de Zacatecas .2,No.3.2013.[En línea].Recuperado en 2020-2-18.Disponible en: https://www.redalyc.org/pdf/5122/512251564005.pdf

ejemplo, se puede demostrar la especificación formal tiene ciertas propiedades de seguridad requeridas, donde el código fuente está libre de errores en tiempo de ejecución y que el código fuente implementa, correctamente las propiedades clave de la especificación. Garantiza que cada paso un refinamiento de los requisitos del usuario en cuanto, el código.

Desarrollo incremental

- Con esta característica se pretende reducir las brechas semánticas entre cada uno de los elementos que hacen parte del proceso con el fin de reducir tiempos con productos terminados.

Evitar la repetición

- Evitar la repetición de los procesos hace que CbyC sea una metodología de alto nivel, dado que estas repeticiones ocasionan primero que todo desgaste y costos para las compañías.

Luchando por la simplicidad

- La lucha por la simplicidad es un trabajo duro, pero vale la pena el esfuerzo. Los métodos de desarrollo de software cada vez más complejos debido que el uso irreflexivo de la orientación a objetos es el principal sospechoso en esta área. La mayor velocidad de producción obtenida mediante el uso de estructuras de datos de biblioteca preconstruidas, fábricas de objetos y asistentes se anulará cuando se intente demostrar la idoneidad para un propósito.

La gestión del riesgo

- La gestión del riesgo debe realizarse caso por caso de una forma sistemática debido que se debe definir los posibles riesgos que se puedan presentar en cualquier fase de desarrollo.

- CbyC enfatiza que el razonamiento lógico debería demostrar la idoneidad para el propósito de un sistema. Un sistema desarrollado de esta manera debe ser certificable a cualquier estándar de seguridad aplicable. Una mentalidad de "caja marcada" centrada en hacer cosas específicas requeridas en un documento de proceso de desarrollo estándar o prescriptivo no logrará el resultado deseado.

5.1.4 Verificación de evaluaciones de seguridad

Dentro del estándar ISO/IEC 15408, está dividido en tres partes, introducción, modelo general, requisitos funcionales y los requisitos de garantía, cada parte se enfoca a un rol que hace parte del ciclo de vida de desarrollo.

Dentro de la parte 3 hace referencia los requisitos de garantía de seguridad a estos se expresa en un perfil protección (PP) o un objetivo de seguridad (ST). Se describe como un conjunto de componentes. Se define los criterios de evaluación del PP y ST donde se estipula los niveles de evaluación, bajo una escala predefinida para definir la evaluación de garantía de niveles (EALs).

Perfil de Protección (PP): según Sogeti[102], se define como una estructura formal e independiente para realizar implementaciones concretas especificadas dentro de un entorno.

A continuación, se describe los criterios para la evaluación de PP:

- Descripción del TOE (APE_DES)
- Ambiente de seguridad (APE_ENV).
- Introducción del PP (APE_INT).
- Objetivos de seguridad (APE_OBJ).
- Requerimientos de seguridad de TI (APE_REQ).
- Declaración explicita de los requerimientos de seguridad de TI(APE_SRE).

Objetivo de Seguridad (ST) Esta estructura solo se aplica a un producto específico, mediante ST un desarrollador podrá identificar que requisitos que satisfacen con la necesidad del producto.

A continuación, se describe los criterios para la evaluación de ST:

- Descripción del TOE (ASE_DES)
- Ambiente de seguridad (ASE_ENV)
- Introducción del ST(ASE_INT)

[102] MELLADO, D.;FERNANDEZ,E y PIATTINI,M.Complementando a métrica: proceso de ingeniería de requisitos de seguridad para el desarrollo de sistemas de información seguros, En: Tecnimap Sevilla.12,No.1.2006.[En línea].Recuperado en 2020-1-12.Disponible en: https://s3.amazonaws.com/academia.edu.documents/57612046/complementando_a_metrica.pdf.

- Objetivos de seguridad (ASE_OBJ)
- Solicitudes de PP(ASE_PPC)
- Requerimientos de seguridad de TI(ASE_REQ)
- Declaración explicita de los requerimientos de seguridad de TI(ASE_SRE)
- Resumen de especificaciones del TOE (ASE_TSS.

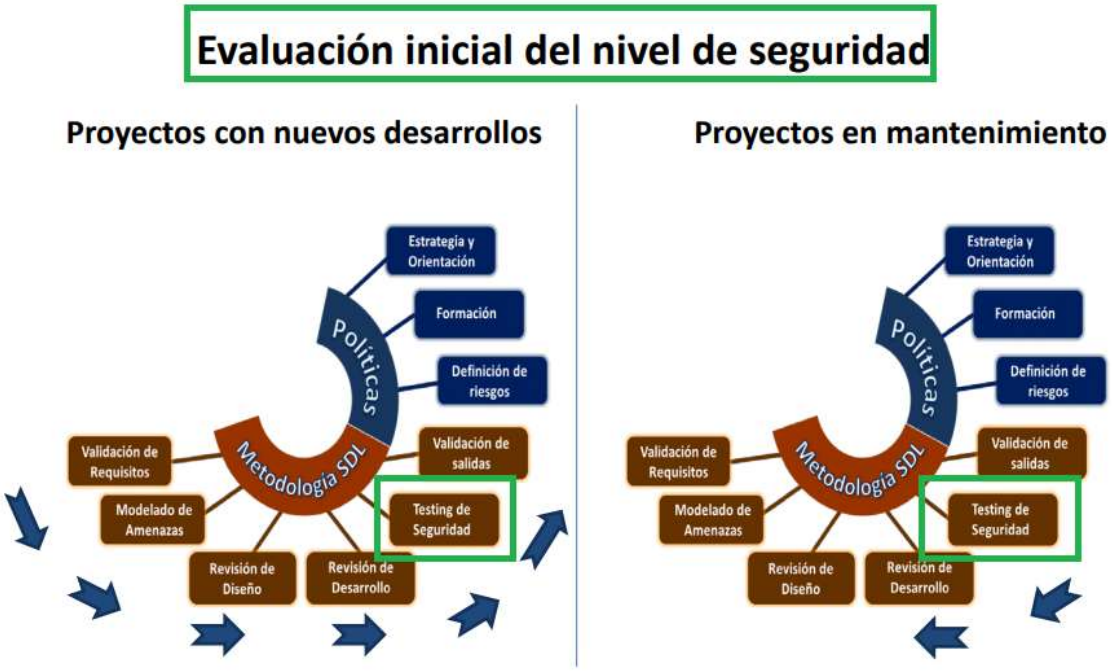
5.1.4.1 Metodologías que aplican evaluaciones de seguridad

En este punto se listará las metodologías de desarrollo de software seguro dentro del ciclo de vida que realiza evaluaciones de seguridad.

- Metodología Security Development Lifecycle (SDL)
- Metodología oracle software security assurance (OSSA)

En la Figura 13, dentro de las actividades que se realiza la metodología **SDL** se evidencia que se realiza evaluaciones de seguridad, tanto para proyectos nuevos como de mantenimiento.

Figura 13. Evaluación de seguridad SDL



Fuente: CARO, Andrés. Evaluación de seguridad SDL,2014.[Imagen digital].Gráfica digital.(14.02x23.1cm),Catedra viewnext, Repositorio Institucional Recuperado de: http://web.fdi.ucm.es/posgrado/conferencias/AndresCaroLindo-slides.pdf el 01 de marzo de 2020

Este proceso se tipifica a través de niveles de validación, como se evidencia en la Figura 14, donde el nivel 2 corresponde con la validación critica con la realización de actividades que se acoplen a un marco regulatorio.

Figura 14.Niveles de validación

Fuente CARO, Andrés. Niveles de evaluación,2014. [Imagen digital]. Gráfica digital.(14.24x22.38cm),Catedra viewnext, Repositorio Institucional Recuperado de: http://web.fdi.ucm.es/posgrado/conferencias/AndresCaroLindo-slides.pdf el 01 de marzo de 2020.

Metodología Oracle Software Security Assurance (OSSA)

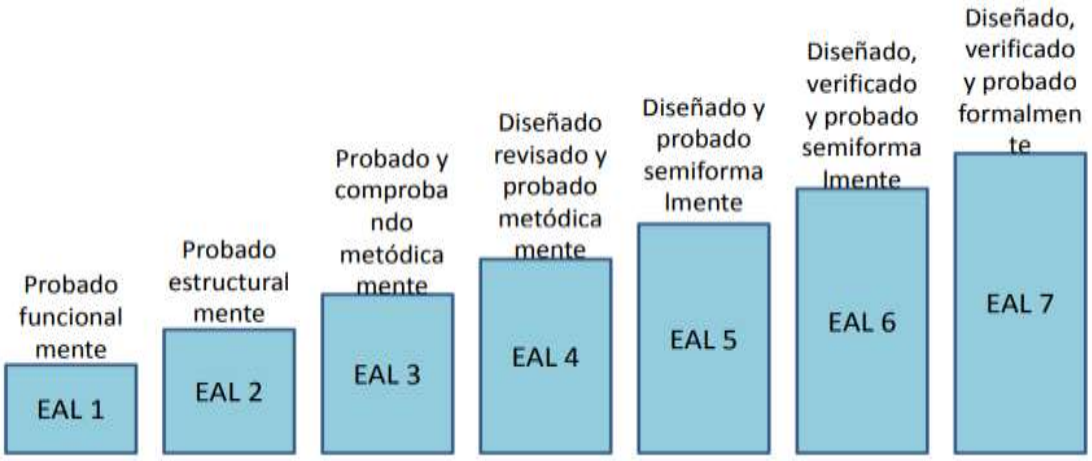
Las evaluaciones de seguridad como FIPS 140-2[103], brindan a los clientes una garantía adicional de que los productos Oracle cumplen con los estrictos requisitos para el procesamiento de datos críticos. Gracias a los productos Oracle evaluados por centros de prueba externos acreditados, los clientes pueden ayudar a cumplir con la creciente de requisitos normativos que se aplican a sus complejos entornos informáticos.

[103] MIN,L y CHEN,T.Analysis of FIPS 140-2 Test and Chaos-Based Pseudorandom Number Generator,En:Revista Chilena de Ingeniería.8,No.1.2013.[En línea].Recuperado en 2020-1-8.Disponible en: http://www.cmsim.eu/papers_pdf/april_2013_papers/4_CMSIM_Journal_2013_Min_Chen_Zang_2_273-280.pdf

5.1.4.2 Niveles de evaluación estándar (ISO/IEC 15408)

En la Figura 15, se enuncia los niveles EALs[104] como un escalafón de evaluaciones, bajo un marco de seguridad establecido dentro de una organización, este se divide en 7 niveles categorizados desde la baja confianza hasta la alta confianza.

Figura 15.Niveles de evaluación de seguridad



Fuente: MELLADO,Daniel.A common criteria based security requirements engineering process for the development of secure information systems,2007.[Imagen digital].Gráfica digital.(10.95x23.13cm),.Revista de investigación. Recuperado de: https://www.sciencedirect.com/science/article/abs/pii/S0920548906000511 el 02 de marzo de 2020.

- EAL1: Funcionalmente a prueba: En este nivel se aplica en el caso que se requiera, esta evaluación a este nivel debe garantizar una evidencia de cada una de las funciones que hacen parte del objeto de evaluación sean consistentes y que sean soportadas con la debida documentación.

- EAL2: Estructuralmente a prueba: Este nivel se requiere que haya una cooperación de los ingenieros de desarrollo, al momento de realizar

[104] SEUNG KOU, K.;JAE GOO,J y GANG SOO,L.Definition of Evaluation Assurance Levels and Estimation of Evaluation Efforts for Operational System Based ISO/IEC 19791, En:International Conference on Security Technology.1,No.1.2008.[En línea].Recuperado en 2020-1-12.Disponible en: https://doi.org/10.1109/SecTech.2008.41.

una distribución que compete a la información del diseño los resultados de las pruebas.

- EAL3: Metódicamente probado y comprobado: En este nivel el desarrollador podrá lograr un máximo nivel de garantía dentro de la seguridad.

- EAL4: Metódicamente diseñados, probados y revisados: Se logra alcanzar un nivel máximo de seguridad con base de buenas prácticas de desarrollo comercial, de las cuales son muy rigurosas y no requiere de un conocimiento especializado.

- EAL5: Semi-formalmente diseñados y probados: Se logra alcanzar el nivel máximo seguridad, con ayuda de una aplicación moderada de técnicas de ingeniería, con base en la búsqueda de vulnerabilidades permite asegurar que el producto resiste ante cualquier ataque de penetración.

- EAL6: Semi-formalmente verificado el diseño y la prueba: Dentro del entorno de evaluación genera un gran valor en cuanto la estimación de los bienes contra los riesgos significativos, además se puede aplicar a aquellos objetos de evaluación, destinados a salvaguardar la seguridad de la informática en las situaciones cuando se presente un riesgo alto.

- EAL7: Formalmente verificado el diseño y la prueba: Esta se aplica cuando se desarrolla con objetos de evaluación, para hacer uso de esta técnica se requiere un alto valor de los bienes donde se supere un alto costo, se puede decir que para aplicar este nivel es de forma ilimitada.

5.1.4.3 Metodologías que cumplen con los 7 niveles de evaluación

Al verificar los niveles de evaluación se encuentra las siguientes metodologías que cumplen con los 7 niveles de evaluación propuestos en el Common Criteria (ISO/IEC 15408):

En primer lugar se encuentra la metodología CLASP, debido esta metodología se descompone en 5 actividades que se pueden distribuir en los 7 niveles,

homologando al elemento CLASP Best Practices, debido que contiene 7 actividades de seguridad como programas de sensibilización, evaluaciones de rendimiento a las aplicaciones, obtención de requerimientos de seguridad, Implementación de buenas prácticas de desarrollo, construcción de planes de solución de vulnerabilidades y la publicación de documentación a través de guías de funcionamiento de seguridad.

En segundo lugar, se encuentra la metodología **SDL**, dentro del proceso de ciclo de vida se puede contar con 7 etapas (entrenamiento, requerimientos, diseño, implementación, verificación, lanzamiento, respuesta) y cada una con 17 practicas, donde se en cada etapa se podrá identificar cada uno de los niveles de evaluación descritos anteriormente.

Con el desarrollo de este capítulo, se logró realizar una recopilación de información, donde se logró identificar características de siete metodologías para desarrollo seguro de las cuales, en el siguiente capítulo serán el objeto de estudio para identificar los criterios a comparar con base al estándar Common Citeria ISO 15408.

Cada una de estas metodologías, demostraron que sus procesos se encuentran en un nivel alto de madurez, donde en cualquier marco de trabajo se pueden adaptar, reduciendo costos y reprocesos.

5.2 REALIZACIÓN DE UN ANÁLISIS COMPARATIVO ENTRE LAS METODOLOGÍAS IDENTIFICADAS BAJO EL ESTÁNDAR ISO/IEC 15408

Se han desarrollado estándares, modelos o directrices que se enfocan en el desarrollo seguro, como es el estándar ISO/IEC 15408, con base en este estándar, las metodologías proporcionan orientación para áreas particulares como el modelado de amenazas, la gestión de riesgos, la codificación segura, pruebas de seguridad, verificación, gestión de parches, gestión de la configuración con el apoyo de evaluaciones de validación en cada fase del desarrollo por consiguiente, en el capítulo anterior se realizó una recopilación de información con el fin de, identificar siete metodologías para el desarrollo de software seguro, junto con sus principales características, punto de partida para realizar un análisis comparativo que se desarrollará en este capítulo.

5.2.1 Definición de los criterios de comparación

Antes de iniciar el respectivo análisis, es necesario establecer los criterios o características que se van a comparar con cada una de las metodologías que fueron investigadas teniendo en cuenta los elementos establecidos en el estándar ISO/IEC 15408.

- Declaración de seguridad (ST): Se define como un conjunto de requisitos de seguridad, donde se establece por referencia a través de una definición de perfiles de protección referenciados a través de normas funcionales.

- Expresión de los requisitos: Este criterio hace referencia a definir un conjunto de estructuras que combinan en grupos de requisitos de seguridad, organizados de forma jerárquica por clase o familia.

- Evaluación de vulnerabilidades: Se enfoca específicamente en aquellas vulnerabilidades que se presentan al momento de la construcción, operación, mal uso de la configuración del objeto de evaluación.

- Evaluación del perfil de protección: Este criterio consiste definir técnicamente como se va usar una declaración de requisito y como aplicarlo dentro un objetivo evaluable (TOE).
- TOE a evaluar: Se debe permitir como definir una evaluación a la gestión de requisitos.

- Especificar límites físicos: Dentro del proceso de la metodología se debe permitir establecer unos límites físicos al momento de realizar la evaluación de los objetivos (TOE).

- Especificar límites lógicos: En este criterio es necesario que se establezca unos límites lógicos que se van a evaluar dentro de una serie de funciones ya definidas.

- Niveles de seguridad de evaluación: Se verifica si las metodologías seleccionadas dentro del proceso del ciclo de vida aplican los niveles de seguridad.

- Evaluación de seguridad: Este criterio se debe cumplir es necesario que cada fase del ciclo de vida se realice un diagnóstico de las actividades ejecutadas.

5.2.2 Identificación de las metodologías

A continuación, se identifican las metodologías ya consultadas de las cuales se procederá a realizar el respectivo análisis comparativo:

- Metodología Security Development Lifecycle (SDL)
- Metodología OPENSAMM
- Metodología BSIMM2
- Metodología CLASP
- Metodología Oracle Software Security Assurance (OSSA)
- Metodología Team Software Process (TSP)
- Metodología Correctness By Construction (CBYC)

5.2.3 Realización del análisis comparativo

Teniendo en cuenta los criterios que se tomaron bajo el estándar ISO 15408, se procede a elaborar el análisis entre las metodologías de desarrollo seguro investigadas en capitulo anterior como se describe en la Tabla 1.

Tabla 1.Análisis comparativo con base en el estándar ISO/IEC 1508

Criterio Common Criteria (ISO/IEC 15408)	SDL	OPENSAMM	BSIMM2	CLASP	OSSA	TSP	CBYC
Declaración de seguridad (ST)	X					X	
Expresión de los requisitos	X	X	X		x	X	X
Evaluación de vulnerabilidades	X	X	X	X			
Evaluación del perfil de protección	X			X	X		
TOE a evaluar						X	
Especificar límites físicos							
Especificar límites lógicos							
Niveles de seguridad de evaluación	X			X	x		
Evaluación de seguridad	X						

Fuente NTC 1486.Analisis comparativo entre metodologías para el desarrollo seguro. Duitama. Universidad abierta y distancia, 2020

Con la identificación de los criterios permitió, realizar el análisis comparativo entre siete metodologías para el desarrollo de software seguro nos da como resultado que la metodología que cumplió con la mayoría de los criterios es la SDL, con base al resultado se puede decir que esta metodología de desarrollo seguro cumplió con estándar (ISO/IEC 15408) por consiguiente, se puede aplicar en cualquier organización.

5.3 REALIZACIÓN DE UN MANUAL PARA EL DESARROLLO DE SOFTWARE SEGURO BAJO EL ESTÁNDAR COMMON CRITERIA (ISO/IEC 15408) Y TAMAÑO DE GRUPOS DE DESARROLLO

En este capítulo se realizará un análisis de cada una de las metodologías que se identificaron en los capítulos anteriores donde sobresale un criterio para tener en cuenta, cuando se desee implementar una metodología de desarrollo seguro dentro de un marco de trabajo, como es el tamaño de los grupos de trabajo. En el capítulo 6.1., se identificaron los procesos y características, de las cuales hay metodologías muy robustas que requieren que se implementen en grupos grandes de desarrollo, debido a su volumen de documentación que se debe levantar en cada proceso.

Finalmente, en este capítulo realizara un manual de buenas prácticas que todos desarrolladores deberían aplicar al momento de desarrollar un programa.

5.3.1 Identificación del tamaño de los grupos de desarrollo

La definición de los desarrolladores depende directamente proporcional del proyecto, debido que en un proyecto con pocos desarrolladores será fácil de identificar cada una de la etapas que se deben realizar, pero si el equipo de desarrollo es numeroso, se deberá analizar cada uno de los factores y estrategias que permitan optimizar tiempo, recurso humano, los productos que cumplan con los estándares de calidad, seguridad, esto puede inducir el uso de metodologías para el desarrollo de software seguro[105].

[105] GUERRERO, C.;SUAREZ,J y GUTIERRES,L.Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web, En:Información tecnológica.24,No.3.2013.[En línea].Recuperado en 2020-14-103.Disponible en: https://doi.org/10.4067/S0718-07642013000300012.

Según Welicki[106], menciona que si se da una solución a corto plazo a un problema específico, pueden colapsar un proyecto.

Dentro de la clasificación de los proyectos se definen de acuerdo con el tamaño y su criticidad:

De acuerdo con el **tamaño** se le determina un valor en función del número de personas que harán parte del desarrollo del proyecto, por general se definen valores dentro de las siguientes secuencias donde estas pueden variar dependiendo la escala Cockburn[107].

- 6: Proyectos entre 1 y 6 ingenieros
- 20: Proyectos entre 7 y 20 ingenieros
- 40: Proyectos entre 21 y 40 ingenieros
- 100: Proyectos entre 41 y 100 ingenieros
- 200: Proyectos entre 101 y 200 personas

De acuerdo con la **criticidad** se define un criterio de acuerdo con lo peor de la situación que se pueda presentar dentro de un fallo del sistema.

- L: fallecimiento
- E: Daños económicos que comprometen con la continuidad del negocio
- D: Daño económico pero que no comprometen con la continuidad del negocio.

Para establecer un grado de formalidad, de acuerdo con la escala Cockburn se diseña una matriz bidimensional[108].

L6L20L40L100L200
E6E20E40E100E200
E6E20E40E100E200
D6D20D40D100D200

[106] WELICKI,L.Patrones y Antipatrones: una Introducción - Parte II,En:microsoft.1,No.1.2008.[En línea].Recuperado en 2020-12-50.Disponible en: https://docs.microsoft.com/es-es/previous-versions/bb972251(v%3dmsdn.10)

[107] DINGOSOYR,T.Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation,En:Empirical Software Engineering.23,No.1.2018.[En línea].Recuperado en 2020-490-520.Disponible en:https://doi.org/10.1007/s10664-017-9524-2.

[108] KETTUNEN,P y LAANTI,M.Combining agile software projects and large-scale organizational agility,En: Software Process: Improvement and Practice.13,No.2.2008.[En línea].Recuperado en 2020-93-183.Disponible en:https://doi.org/10.1002/spip.354

De acuerdo con el grado de formalidad, se podrá determinar la metodología para el desarrollo de software seguro, puede darse un caso en que exista diferentes metodologías que cumplan para un grupo concreto en cuanto tamaño y criticidad. Cockburn después de muchos años de experiencia formulo unos criterios que se deben tener en cuenta para seleccionar una metodología adecuada.

- Comunicación interactiva

Permite ser una canal rápido y barato para intercambiar información, debido que permite la reducción de costos dentro del proyecto, donde facilita el desarrollo a mayor tamaño el proyecto dificulta la comunicación de este tipo.

- El exceso en la metodología pesada es muy costoso

Debido a la gran cantidad de documentación y artefactos, que por características del proyecto pueden que no sean necesarias sino al contrario entorpecen con el enfoque en las actividades, donde en realidad generan valor.

- Equipos grandes necesitan metodologías pesadas:

Al utilizar una metodología liviana para un equipo de desarrollo mayor de 10, no es viable debido que la comunicación se convierte en un dolor de cabeza y dentro de un mismo ambiente, por consiguiente, se necesita revisar que metodologías, permiten una comunicación efectiva, para no caer en los excesos de actividades[109]. De acuerdo con el punto anterior se puede identificar la clasificación de proyectos de acuerdo con el número de ingenieros, se define el proyecto si es grande o pequeño, luego de haber realizado esta identificación es necesario saber cómo se encuentran clasificadas las compañías con base en el número de desarrolladores[110].

[109] WANG,F.The Best-of-2-Worlds Philosophy: Developing Local Dismantling and Global Infrastructure Network for Sustainable e-Waste Treatment in Emerging Economies,En:Waste Management, Special Thematic Issue: Waste Management in Developing Countries.32,No.11.2012.[En línea].Recuperado en 2020-46-2134.Disponible en:https://doi.org/10.1016/j.wasman.2012.03.029.

[110] MIRNA,M.;GASCA,G y VALTIERRA,C.Caracterizando las Necesidades de las Pymes para Implementar Mejoras de Procesos Software: Una Comparativa entre la Teoría y la Realidad, En:RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação.1,No.15.2014.[En línea].Recuperado en 2020-1-15.Disponible en: https://doi.org/10.4304/risti.e1.1-15.

Con base en una categorización[111] genérica para las pymes como se puede mostrar en la siguiente Tabla 2, se identifica la cantidad de ingenieros por cada categoría.

Tabla 2.Clasificación de Pymes para empresas de desarrollo de software

Categoría Pyme	Numero ingenieros
Mediana	51 a 130
Pequeña	10 a 50
Micro	1 a 9

Fuente CARDOZO, D.; VELASQUEZ, Y RODRIGUEZ, C. Revisión de la definición de PYME en América Latina. Latin American, Latin American and Caribbean Conference for Engineering and Technology. Megaprojects.2012,10p.

5.3.2 Elaboración del manual de desarrollo seguro

En este punto se describe cual es la metodología de desarrollo seguro, podría adoptar las compañías teniendo como criterio base el número de personas, puesto que no todas las metodologías se pueden aplicar a cualquier grupo de desarrolladores, debido al gran número de insumos que se tienen que levantar en cada fase. Por lo tanto, si no se escoge la metodología correcta de desarrollo seguro, esto acarraría costos innecesarios.

Pyme micro

En este grupo puede tener un grupo máximo de 9 empleados las siguientes metodologías de desarrollo seguro se puede aplicar:

Metodología Team Software Process (TSP): Esta metodología se puede aplicar a las empresas de tipo Pyme Micro porque se produce software seguro y fiable en menos tiempo, con pocos desarrolladores y con un costo menor, ya que se realiza un balance enfocado en un proceso de construcción de proyectos pequeños que no requieran un numero grande de ingenieros y la documentación es liviana. Los grupos de trabajo no necesariamente tienen que ser de alto nivel de rendimiento, son autónomos de establecer metas y son dueños de sus procesos y planes.

[111] CARDOZO,D.;VELASQUEZ,Y y RODRIGUEZ,C.Revisión de la definición de PYME en América Latina, En:10th Latin American and Caribbean Conference for Engineering and Technology. Megaprojects.10,No.1.2012.[En línea].Recuperado en 2020-1-10.Disponible en: http://oa.upm.es/19446/.

Oracle Software Security Assurance: Esta metodología se puede aplicar teniendo en cuenta que en estas compañías, ya deben contar con licencias de Oracle dentro de su arquitectura de bases de datos por consiguiente, se convierte un plus que Oracle como una compañía reconocida a nivel mundial, se enfocara en el tema de seguridad desde la arquitectura de las bases de datos, debido que se convierte en un foco de vulnerabilidad, dado que se pueden ejecutar scripts que ocasionen daños en la información existente.

Esta metodología fue diseñada por Oracle, garantizando que se pretenda analizar áreas generales teniendo como objetivo central la seguridad, aplicando principios de diseño, seguridad en comunicaciones, vulnerabilidades comunes y criptografía, donde se proporcionan una orientación especifica con todos los temas que tiene que ver con la validación de datos, privacidad, CGI[112] y finalmente la administración de usuarios.

Pyme pequeña

Este grupo de compañías se puede aplicar las siguientes metodologías de desarrollo seguro:

Metodología correctness by construction(CBYC): Esta metodología se puede aplicar a estas compañías porque manejan aplicaciones donde se necesitan tasas de defectos extremadamente bajas para el software de alta integridad, compuesto por millones de líneas de código.
La clave para la adopción exitosa de CbyC, es la adopción de una mentalidad de ingeniería. En particular las decisiones sobre procesos, métodos y herramientas para el desarrollo de software deben, basarse en base de la lógica y la precisión.
Estas compañías se hablan entre un numero de 10 a 50 desarrolladores se deben enfocar que CbyC aplica rigor a todas las fases de desarrollo de software, incluido el diseño detallado, implementación y verificación, por consiguiente, deben tomar conciencia de la importancia de aplicar esta metodología dentro de sus procesos.

Metodología (CLASP): Esta metodología se puede aplicar dentro de una compañía pequeña, debido que cuenta con una serie de actividades de alto nivel facilitando que la implementación se realice a través de un proceso ligero, que no conlleve mucha documentación, donde beneficiará a las compañías en cuanto costos, debido que solucionara requerimientos en menor tiempo y con estándares de

[112]SPEARING,M.Modification of the Clinical Global Impressions (CGI) Scale for Use in Bipolar Illness (BP) The CGI-BP,En: Psychiatry Research.73,No.3.1997.[En línea].Recuperado en 2020-71-159.Disponible en:https://doi.org/10.1016/S0165-1781(97)00123-6.

calidad y seguridad, adicional los desarrolladores se podrán adaptar de una manera ágil a un nuevo proceso dentro de las fases de desarrollo[113].

.

Los desarrolladores podrán hacer uso de herramientas que permitirán automatizar cada una de las fases del ciclo de vida facilitando que sea un proceso óptimo para realizar revisiones en cuanto las vulnerabilidades o errores que se presenten.

Pyme mediana

Metodología (BSIMM2): Esta metodología se puede aplicar en compañías medianas debido a nivel mundial esta metodología es aplicada por más de 50 compañías reconocidas como Google, Nokia, Sap , Intel, e.t.c.
La documentación es extensa ya que esta metodología recoge las diferentes propuestas que son aplicadas en otras metodologías de desarrollo seguro como (OWASP, CLASP), con el objetivo que las compañías puedan seguirlo paso a paso y así podrán identificar en qué nivel de madurez se encuentran frente al desarrollo seguro.

Por consiguiente, esta metodología se convierte como una herramienta que permite ejecutar métricas de medición de madurez dado que está basado en datos reales, medibles y comparables.

Metodología Security Development Lifecycle (SDL): Teniendo en cuenta que esta metodología es pesada y rigurosa, por consiguiente, se puede aplicar en compañías entre un numero de 51 a 130 empleados. Por consiguiente, se podrá distribuir las actividades de una forma equitativa, con el objetivo de optimizar tiempos de respuesta cumpliendo con los estándares de calidad y seguridad.

Esta metodología se convierte en la más rigurosa, puesto que se debe cumplir con las actividades de seguridad que son obligatorias y se agrupan en cada fase dentro del ciclo de vida.

Al aplicar esta metodología en compañías medianas, se puede anticipar fallas en el código, por consiguiente, se aplican métodos que permiten proteger el código. Es necesario cambiar la mentalidad de los desarrolladores; si se presenta la falla se procede a solucionar de una vez y no dejarlo para después. Adicional, se podrá remover código viejo innecesario que no cumple con ninguna funcionalidad con las nuevas versiones, se debe reemplazar protocolos viejos, cumpliendo con el enfoque de la metodología a la mejora continua.

[113] BROWNING,J.Effect of Positional Loading of Three Removable Partial Denture Clasp Assemblies on Movement of Abutment Teet,En: The Journal of Prosthetic Dentistry.55,No.3.1986.[En línea].Recuperado en 2020-51-347.Disponible en: https://doi.org/10.1016/0022-3913(86)90118-6.

A continuación, se relaciona la metodología que se puede aplicar en cualquier clasificación de Pyme:

Metodología opensamm: Esta metodología se puede aplicar para todo tipo de compañías micro, pequeña y mediana, dado que fue definida para ser flexible sin importar el estilo de desarrollo ya que se puede aplicar en toda la organización, dentro de una sola línea de negocio también un negocio particular.

Durante el proceso en que se desarrolla esta metodología tendrá cambios en cuanto el comportamiento de una organización a través de un lapso, debido que durante el proceso de desarrollo se realizará pequeños entregables con el fin de identificar vulnerabilidades a largo plazo.

La ventaja que tiene esta metodología se puede aplicar en cualquier compañía no se limita a un solo lineamiento, sino al contrario cada empresa que opte este modelo podrá personalizar sus opciones de acuerdo con sus necesidades.

En cuanto los lineamientos relacionados con las actividades de seguridad en cuanto el aseguramiento es simples, bien definidos y se podrá generar métricas.

5.3.3 Definición de buenas prácticas generales

Lineamientos de acuerdo con el estándar common criteria (ISO/IEC 15408)

A continuación, se menciona los siguientes definidos por el autor Zimmermann[114]:

Análisis de los riesgos de acuerdo con los niveles de seguridad (EAL) En este lineamiento se basa en dos niveles como son EAL1, donde en este nivel se deben verificar que cada una de las fases sean probadas funcionalmente, cuando se llegue a revisar, el nivel EAL2, donde en este nivel se debe garantizar las pruebas estructurales dentro de los controles ya definidos dentro de las clases; administración de las configuraciones, entregas y funcionamiento, desarrollo, pruebas y finalmente evaluaciones de vulnerabilidades.

Realizar evaluaciones de seguridad a los productos Como buena práctica de acuerdo con el estándar Common Criteria (ISO/IEC 15408), los desarrolladores deben realizar las evaluaciones de cada uno de los insumos que se encuentran listos para salir a producción, conceptualizados en un objetivo de evaluación (TOE). Con esta buena práctica se podrán identificar las debilidades y formular las respectivas recomendaciones.

[114] ZIMMERMANN,J.OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection,En: IEEE Transactions on Communications.28,No.4.1980.[En línea].Recuperado en 2020-32-425.Disponible en: https://doi.org/10.1109/TCOM.1980.1094702.

Identificación de los limites Es necesario especificar los limes físicos, conceptualizados dentro de los componentes del hardware que permitirá soportar los límites de evaluación.

Especificar los limites lógicos Es importante que cuando se ejecute esta práctica se identifique y se autentique los usuarios antes de acceder al objetivo de evaluación (TOE), ya que se demuestra uso de los controles establecidos dentro de las políticas de seguridad.

Identificar el entorno de seguridad Es necesario que esta práctica se definan las diferentes hipótesis en cuanto el entorno físico, hipótesis del entorno TI, teniendo en cuenta la conexión controlada de sistemas externos al TOE, que interactúen con él.

Tener claro los objetivos del TOE/RTOE Es necesario que los productos desarrollados se establezca los controles de acceso de cualquier usuario que se encuentre autorizado, a través de identificaciones de tipo OS1.

Verificar las funciones de seguridad Es vital importancia que en las etapas de desarrollo se pueda identificar que funciones de seguridad toma de referencia al TOE.

Evaluar las funcionalidades de los niveles de evaluación Dentro de los lineamientos establecidos por el estándar COMMON CRITERIA (ISO/IEC 15408) enfocados al TOE, es necesario establecer en cada fase de desarrollo una realice documentación, donde los niveles de evaluación fueron probados estructuralmente donde se evalúa la posibilidad de hacer entregas de la documentación del diseño y el resultado de las pruebas TOE.

Verificar las evaluaciones de seguridad Se recomienda establecer unos valores de ponderación, es decir unos puntos en cada paso para el cumplimiento de disposiciones en cuanto los límites, objetivos de seguridad y funciones de seguridad. Para las actividades de cumplimiento cinco, tres, dos, disposiciones tres, dos, uno puntos, de acuerdo con el entorno.

Aplicar evaluaciones TOE El enfoque central del estándar COMMON CRITERIA (ISO/IEC 15408), son las evaluaciones de seguridad, por tal motivo es de carácter obligatorio que se apliquen dichas evaluaciones para cada actividad establecida, a través de niveles de cumplimiento (ideal, adecuado, aceptable, regular, critico).

Recomendaciones generales

Una buena práctica ha demostrado excelentes resultados, los autores Futcher y Von Solms[115],describen a continuación las siguientes recomendaciones generales que ayudarán a prevenir y detectar amenazas.

- **Realizar validaciones de entrada**: Es necesario que se realice validaciones de parámetros que vienen ingresados a través de formularios de ingreso de usuario. Estos parámetros deben ser capturados ser manipulados a través de código, validando cada uno de los caracteres que contenga cada parámetro, si no se encuentran en una lista de caracteres permitidos, se debe generar un error en la aplicación cliente.

- **Manejo de sentencias SQL** En caso en que las consultas son construidas directamente con información que provienen del usuario entre líneas o son concatenadas con texto de la consulta, en lo posible no utilizar parámetros que sean de vinculación, debido que se deja una venta para que haya una vulnerabilidad, donde se haga inyección de SQL para evitar este tipo de ataque se recomienda hacer uso de las siguientes líneas de código:

En la siguiente Figura 16, se evidencia un ejemplo en código PHP, de cómo se puede ejecutar sentencias que permitirá proteger el código, ante cualquier ataque por inyección con el uso de prepared statements y parameterized queries (PDO).

Figura 16. Ejecutar sentencias de forma segura

```php
$stmt = $pdo->prepare('SELECT * FROM usuarios WHERE nombre = :nombre');
$stmt->execute(array('nombre' => $nombre));
foreach ($stmt as $row) {
    // Hacer algo con $row
}
```

Fuente: GUAMAN, Daniel. Ejecutar sentencias de forma segura,2017. [Imagen digital].Gráfica digital.(4.05x13.05cm),Universidad Técnica Particular de Loja, Repositorio Institucional. Recuperado de: https://www.researchgate.net/profile/Daniel_Guaman4/publication/318416819_Implementation_of_techniques_and_OWASP_security_recommendations_to_avoid_SQL_and_XSS_attacks_using_J2EE_and_WS-Security/links/5a9771f70f7e9ba42974d52f/Implementation-of-techniques-and-OWASP-security-recommendations-to-avoid-SQL-and-XSS-attacks-using-J2EE-and-WS-Security.pdf el 10 de abril de 2020.

[115]FUTCHER,L y VON SOLMS,R.Guidelines for secure software development, En: Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries.8,No.1.2008.[En línea].Recuperado en 2020-56-65.Disponible en: https://doi.org/10.1145/1456659.1456667.

- **Comentar código** Si se encuentra código documentado, se debe eliminar antes de desplegar un producto a producción debido que en producción se puede alterar el funcionamiento de la misma, también se debe tener en cuenta que tampoco, debe haber código quemado, debido que la mayoría de los desarrolladores deja el código para hacer pruebas, pero por las premuras de las entregas se deja al olvido y no se opta por eliminar el código innecesario.

- **Existencia de contenidos URL** Es de vital importancia tener claro que nunca se debe dejar contenido dentro de las URL, como por ejemplos contraseñas, IP, nombres de servidores, leguajes de programación, rutas de archivos entre otras, puesto a anterior se deja una ventana de inseguridad gravísima.

- **Generación de los mensajes de error** En lo posible evitar que los mensajes que se generan a cara del usuario final, contenido como información de servidores, leguajes de programación, consultas, etc. Se deben estandarizar dentro de un manejo de errores de forma que sean entendibles.

- **Dejar Valores quemados** Se ha identificado que gran parte de los desarrolladores manejan HARDCODE, es decir código quemado por lo general variables definidas por defecto, se puede decir que es una mala práctica dado que dentro de las asignaciones de estas variables se dejan contraseñas, direcciones ip por consiguiente, se convierte una vulnerabilidad debido que un atacante puede aprovechar para conocer el entorno y explotar fallas de seguridad.

- **Manejar privilegios de acceso** Este se convierte en una práctica fundamental que todas las empresas se deben implementar políticas de roles y privilegios, debido que las aplicaciones, deben proporcionar que la información se debe visualizar a los funcionarios que requieran la información necesaria.

- **Establecer que todas contraseñas se encuentren cifradas** Es de carácter obligatorio que las contraseñas utilizadas en código deben contar con un algoritmo de función de resumen SHA-1, por consiguiente, es necesario que las compañías implementen políticas en cuanto el manejo de contraseñas.

En este capítulo se pude concluir que existen metodologías para el desarrollo seguro muy robustas y completas pero no todas se pueden implementar en cualquier tipo de organización, antes de implementar una metodología se debe

realizar un análisis en cuanto el número de personas presentes en cada uno de los equipos de desarrollo, dado que en estas metodologías demanda de varias fases que se debe ejecutar de una manera sincronizada y con un grupo grande de desarrolladores no tan juniors.

Es importante que los desarrolladores tomen conciencia de la importancia de implementar metodologías de desarrollo seguro dado que estas generan un gran valor a las compañías debido que se evita reprocesos y las más importantes vulnerabilidades.

6 CONCLUSIONES

A través de la recopilación de información, se logró identificar las diferentes metodologías para el desarrollo de software seguro debido que esta información fue vital para el desarrollo de las demás fases del proyecto.

Con la identificación de las principales características de cada una de las metodologías se obtuvieron resultados satisfactorios debido que con base en estos elementos se logró realizar un análisis, para identificar que metodologías se adapta de acuerdo con el tamaño de los grupos de desarrollo.

Se identificó la importancia de, implementar evaluaciones de seguridad, como lo exige el estándar Common Criteria (ISO/IEC 15408), en cada fase de desarrollo, debido que se podrá detectar bugs de seguridad en un menor tiempo, garantizando un producto de alta calidad.

Con base en el estándar Common Criteria (ISO/IEC 15408), se logró definir los criterios requeridos con los cuales se realizó un análisis comparativo entre las metodologías de desarrollo seguro, con el fin de identificar cuáles de estas cumplen con requerimientos exigidos por el estándar.

Fue de vital importancia la definición de las metodologías de desarrollo seguro, dado que para realizar el análisis comparativo fue necesario identificar que metodologías de debían comparar.

Gracias al análisis comparativo se logró concluir que, de las siete metodologías seleccionadas para el desarrollo de software seguro, la metodología que cumplió con la mayoría de los criterios de acuerdo con el estándar internacional Common Criteria (ISO/IEC 15408) es; Security Development LifeCycle (SDLC), dado que en todas las fases del SDLC se realiza una evaluación de estado, es decir desde la fase de requisitos hasta la liberación, a través de un proceso de seguimiento.

Con la identificación del tamaño de los grupos de desarrollo, es de gran ayuda debido que antes de implementar una metodología de desarrollo de software seguro, es necesario definir qué tipo de compañía es, de acuerdo al número de desarrolladores, debido que no se puede adaptar cualquier metodología hasta que no se realice un análisis previo dado que existen metodologías, donde sus procesos manejan documentaciones robustas de los cuales requieren un gran número desarrolladores, por consiguiente se debe revisar que metodología se adapta a las necesidades y expectativas de las compañías, con el fin de no incurrir en gastos innecesarios.

7 RECOMENDACIONES

Con la realización de esta monografía, como fue el estudio de siete metodologías para el desarrollo seguro, siempre se desea que haya una mejora continua, debido que día tras día el mundo sigue innovando en crear nuevos modelos de desarrollo seguro de los cuales, se podría continuar con un análisis comparativo; por consiguiente se recomienda para futuros estudios que se tenga en cuenta este trabajo con el fin completar, con más metodologías y así poder realizar un análisis comparativo entre un mayor número de metodologías y poder brindar más opciones a diferentes compañías que deseen implementar metodologías para el desarrollo seguro.

Otra recomendación es especialmente para las compañías de desarrollo, que dentro de sus equipos lideren los procesos de seguridad a través del uso de metodologías para desarrollo seguro, debido que un gran porcentaje de desarrolladores no conocen de modelos y buenas prácticas de seguridad.

Se sugiere que se continúe con la posibilidad de crear e innovar una nueva metodología de desarrollo seguro, debido que día a día se crean nuevos lenguajes de programación donde se requiere que la seguridad sea aplicada al momento de codificar y que se puedan adaptar a marcos de trabajo agiles.

8 BIBLIOGRAFÍA

ALMEIDA, J.;NETO,J y QUEIROS,L.A construção de software seguro: uma análise a partir dos modelos de desenvolvimento e maturidade, En:Revista Eletrônica da Estácio Recife.2,No.2.2016.[En línea].Recuperado en 2020-89-581.Disponible en: https://reer.emnuvens.com.br/reer/article/view/84.

ALMEIDA,G.Legal Rules and Information Security Technical Standards: Possible Approach for Filling in the Blanks of Cybercrime Legislation. SSRN Electronic Journal [En línea].2011,Vol.2Nro.1.[Consultado 20 de febrero de 2020].Disponible en: https://doi.org/10.2139/ssrn.1742962.

AVELT,M ,ANUSHA,P y LOELLA,M.Secure SDLC for IoT Based Health Monitor: A Summary of Recent Results.Second International Conference on Electronics [En línea].2018,Vol.2Nro.1.[Consultado 30 de marzo 2020].Disponible en: https://doi.org/10.1109/ICECA.2018.8474668.

BACA,D.Y Bengt,C.Agile Development with Security Engineering Activities.[En línea].3°.ed.(Proceeding of the 2nd Workshop on Software Engineering for Sensor Network Applications;nro.11)USA:Edimat,2011.[Citado el 11 de diciembre del 2019].Disponible en: https://doi.org/10.1145/1987875.1987900.

BALICA, F, WRIGHT, G, Y MEULEN,F.A Flood Vulnerability Index for Coastal Cities and Its Use in Assessing Climate Change Impacts. Natural Hazards [En línea].2012,Vol.64Nro.1.[Consultado 28 de diciembre 2019].Disponible en: https://doi.org/10.1007/s11069-012-0234-1
IDC COLOMBIA. [sitio web]. Bogotá:IDC,Analiza el futuro.[Consulta: 16 de febrero 2020].Disponible en:http://www.idccolombia.com.co/.

BARCELL, M.Estudio de una estrategia para la implantación de los sistemas de gestión de la seguridad de la información, , En: Universidad de Cadiz.2,No.3.2003.[En línea].Recuperado en 2020-2-18.Disponible en: http://www.mfbarcell.es/conferencias/Metodolog%C3%ADas%20de%20seguridad_2.pdf.

BERNSMED, K .;GILIE,M y HAKON,P.Safety Critical Software and Security - How Low Can You Go?, En: IEEE/AIAA 37th Digital Avionics Systems Conference.37,No.1.2018.[En línea].Recuperado en 2020-1-6.Disponible en: https://doi.org/10.1109/DASC.2018.8569579.

BRITO, Metodologías para desarrollar software seguro, En: Universidad Autónoma de Zacatecas.2, No.3.2013. [En línea]. Recuperado en 2020-2-18. Disponible en: https://www.redalyc.org/pdf/5122/512251564005.pdf.

BROWN, A.Realizing service-oriented solutions with the IBM Rational Software Development Platform, En:IBM.44,No.4.2005.[En línea].Recuperado en 2020-727-52.Disponible en: https://doi.org/10.1147/sj.444.0727.

BROWN, M y PALLER,A.Secure Software Development: Why the Development World Awoke to the Challenge, En: Information Security Technical Report.13,No.1.2008.[En línea].Recuperado en 2020-40-43.Disponible en: https://doi.org/10.1016/j.istr.2008.03.001.

BROWNING,J.Effect of Positional Loading of Three Removable Partial Denture Clasp Assemblies on Movement of Abutment Teet,En: The Journal of Prosthetic Dentistry.55,No.3.1986.[En línea].Recuperado en 2020-51-347.Disponible en: https://doi.org/10.1016/0022-3913(86)90118-6.

BURNSTEIN, I.;SUWANNASSART,T y CARLSON,R.Developing a Testing Maturity Model for software test process evaluation and improvement, En:Proceedings International Test Conference.1,No.3.1996.[En línea].Recuperado en 2020-89-581.Disponible en: https://doi.org/10.1109/TEST.1996.557106.

CADAVID, A. Revisión de metodologías ágiles para el desarrollo de software.Prospectiva [En línea].2013,Vol.11Nro.2.[Consultado 20 de noviembre de 2019].Disponible en:https://doi.org/10.15665/rp.v11i2.36.

CAMACHO,J.Prototipo Web para la Gestión de Métricas como Soporte a la Metodología de Desarrollo TSP ,En: Repositorio Institucional Universidad Distrital - RIUD .6,No.1.2017.[En línea].Recuperado en 2020-309-401.Disponible en: http://repository.udistrital.edu.co/handle/11349/6236.

CAMPOS, S y ROLANDO,J.Implementación de la metodología Six Sigma SDLC para la mejora de la calidad del proceso de desarrollo web,En: Repositorio UnaAcCr.1,No.1.2015.[En línea].Recuperado en 2019-94-140.Disponible en:https://repositorio.una.ac.cr/handle/11056/10550.

CARDOZO, D.; VELASQUEZ y RODRIGUEZ, Revisión de la definición de PYME en América Latina, En:10th Latin American and Caribbean Conference for Engineering and Technology. Megaprojects.10, No.1.2012. [En línea]. Recuperado en 2020-1-10. Disponible en: http://oa.upm.es/19446/

CARRIZO, D y ALFARO,A.Método de Aseguramiento de La Calidad En Una Metodología de Desarrollo de Software: Un Enfoque Práctico,En:Revista Chilena de Ingeniería.26,No.1.2018.[En línea].Recuperado en 2020-29-114.Disponible en: https://doi.org/10.4067/S0718-33052018000100114.7.

CHAMORRO, J y PINO,F.Modelo para la evaluación en seguridad informática a productos software, basado en el estándar ISO/IEC 15408 Common Criteria, En: Sistemas y Telemática.9,No.19.2011.[En línea].Recuperado en 2020-69-92.Disponible en: https://doi.org/10.18046/syt.v9i19.1095.

CHANDRA, P.Software Assurance Maturity Model (SAMM) a guide to building security into software development, En:OpenSamm Project Lead.1,No.1.2009.[En línea].Recuperado en 2020-15-38.Disponible en: http://www.cs.unh.edu/~it666/reading_list/SDLC/opensamm_1.0.pdf.

DAVIS,N.Processes for producing secure software, En:IEEE Security & Privacy Magazine.2,No.3.2004.[En línea].Recuperado en 2020-897-911.Disponible en:https://doi.org/10.1109/MSP.2004.21.

DE WIN, B.; SCANDARIATO,R.;BUYENS,K,.;GREGORIE,J y WOUNTER,J .On the secure software development process CLASP, Network Security.51,No.7.2009.[En línea].Recuperado en 2009-1152-1171.Disponoble en:https://www.sciencedirect.com/science/article/abs/pii/S0950584908000281.

DE WIN,B.On the Secure Software Development Process: CLASP, SDL and Touchpoints Compared,En: Information and Software Technology.51,No.7.2009.[En línea].Recuperado en 2019-71-1152.Disponible en: https://doi.org/10.1016/j.infsof.2008.01.010.

DINGOSOYR,T.Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation,En:Empirical Software Engineering.23,No.1.2018.[En línea].Recuperado en 2020-490-520.Disponible en:https://doi.org/10.1007/s10664-017-9524-2.

DONG, G.Concepto, Security Assurance with Metamorphic Testing and Genetic Algorithm, En:IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.1,No.4.2010.[En línea].Recuperado en 2020-397-401.Disponible en: https://doi.org/10.1109/WI-IAT.2010.101.

DONGGILL, L. A New Integrated Software Development Environment Based on SDL, MSC, and CHILL for Large-Scale Switching Systems,En:ETRI Journal.18,No.4.1997.[En línea].Recuperado en 2020-86-265.Disponible en: https://doi.org/10.4218/etrij.97.0197.0044.

ENCK, W. A Study of Android Application Security, En: SYSTEMS AND INTERNET INFRASTRUCTURE SECURITY.1, No.4.2011. [En línea]. Recuperado en 2020-3-38.Disponible en: https://www.usenix.org/legacy/event/sec11/tech/slides/enck.pdf.

FAN, R y CHANG,Y.Machine Learning for Black-Box Fuzzing of Network Protocols, En: Information and Communications Security, ed. Sihan Qing et al., Lecture Notes

in Computer Science.9,No.19.2018.[En línea].Recuperado en 2020-621-32.Disponible en: https://doi.org/10.1007/978-3-319-89500-0_53.

FERNANDEZ, E y ASTUDILLO,H.Experimental evaluation of secure software methodologies using patterns, En:Acm dl Digital Library.16,No.11.2020.[En línea].Recuperado en 2020-1-7.Disponible en:https://dl.acm.org/doi/10.5555/3124362.3124368.

FUTCHER,L y VON SOLMS,R.Guidelines for secure software development, En: Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries.8,No.1.2008.[En línea].Recuperado en 2020-56-65.Disponible en: https://doi.org/10.1145/1456659.1456667.

GEFEI, S.A supporting tool for creating and maintaining security targets according to ISO/IEC 15408, En: IEEE International Conference on Computer Science and Automation Engineering
.3, No.3.2012.[En línea].Recuperado en 2020-745-49.Disponible en: https://doi.org/10.1109/ICSESS.2012.6269574.

GEPPERT, B y ROBLER,F.The SDL Pattern Approach – a Reuse-Driven SDL Design Methodology,En: Computer Networks .35,No.6.2001.[En línea].Recuperado en 2020-45-627.Disponible en: https://doi.org/10.1016/S1389-1286(00)00202-4.

GONZALEZ,D.La protección de información y los datos en el marco de la ley 1273 de 2009: un estudio del dato y la información como objeto material en el tipo penal hurto por medios informáticos, En: Universidad La Gran Colombia.1,No.1.2017.[En línea].Recuperado en 2020-5-78.Disponible en: http://repository.ugc.edu.co/handle/11396/4273.

GREGOIRE.On the Secure Software Development Process: CLASP and SDL Compared.Third International Workshop on Software Engineering for Secure Systems [En línea].2007, Vol.1Nro.1.[Consultado 11 de noviembre 2019].Disponible en: https://doi.org/10.1109/SESS.2007.7.

GUERRERO, C.; SUAREZ, J y GUTIERRES,L.Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web, En:Información tecnológica.24,No.3.2013.[En línea].Recuperado en 2020-14-103.Disponible en: https://doi.org/10.4067/S0718-07642013000300012.
HALL,A,CHAPMAN,R.Correctness by construction: developing a commercial secure system.IEEE Software [En línea].2002,Vol.19Nro.1.[Consultado 15 de diciembre 2019].Disponible en:https://doi.org/10.1109/52.976937.

HANSSON,S.Risk and Safety in Technology,En: Philosophy of Technology and Engineering Sciences.1,No.1.2009.[En línea].Recuperado en 2019-1069-1102.Disponible en: https://doi.org/10.1016/B978-0-444-51667-1.50043-4.

HERNANDEZ,Antonio.Sistema para la detección de ataques PHISHING utilizando correo electrónico.Revista Telemática 17.2019,nro.2.pp.60-70.ISSN 0120-3916.

HERZOG,Felix.Straftaten im Internet, Computerkriminalitat und die Cybercrime Convention.Revista política criminal 4.2019,nro.8.pp.475-84.ISSN 0120-3916.
HILBRICH,R.How to Safely Integrate Multiple Applications on Embedded Many-Core Systems by Applying the Correctness by Construction Principle,En:Advances in Software Engineering.2,No.3.2012.[En línea].Recuperado en 2020-1-14.Disponible en:https://doi.org/10.1155/2012/354274.

HOGAN, J.; SMITH, G y THOMAS, R.The Real World Software Proces, En: Ninth Asia-Pacific Software Engineering Conference.29,No.2.200.[En línea].Recuperado en 2020-366-75.Disponible en: https://doi.org/10.1109/APSEC.2002.1183006.

HORIE, D.; MORIMOTO,S y CHENG,J.A Web User Interface of the Security Requirement Management Database Based on ISO/IEC 15408, En:Computational Science – ICCS 2006, ed. Vassil N.1,No.1.2006.[En línea].Recuperado en 2020-797-804.Disponible en: https://doi.org/10.1007/11758549_107.

HSUEH, N.Concepto, Applying UML and Software Simulation for Process Definition, Verification, and Validation, En:Information and Software Technology.50,No.9.2008.[En línea].Recuperado en 2020-897-911.Disponible en: https://doi.org/10.1016/j.infsof.2007.10.015

HUMPHEREY, W.Team Software Process (TSP), En:Encyclopedia of Software Engineering American Cancer Society.1,No.1.2002.[En línea].Recuperado en 2020-897-911.Disponible en:https://doi.org/10.1002/0471028959.sof352.
INFOSPYWARE. [sitio web]. Bogotá:CEO,¿Qué es el Phishing?.[Consulta: 9 de abril 2020].Disponible en:https://www.infospyware.com/articulos/que-es-el-phishing/.

KELLEY, D.Measuring Software Security: BSIMM2 and Beyond,eSecurty, En: eSecurty Planet .37,No.1.2010.[En línea].Recuperado en 2020-1-6.Disponible en: https://www.esecurityplanet.com/views/article.php/3881771/Measuring-Software-Security-BSIMM2-and-Beyond.htm.

KETTUNEN, P y LAANTI,M.Combining agile software projects and large-scale organizational agility,En: Software Process: Improvement and Practice.13,No.2.2008.[En línea].Recuperado en 2020-93-183.Disponible en:https://doi.org/10.1002/spip.354

KIM, H.; WEN, J y VILLASENOR,J.Secure Arithmetic Coding, En:IEEE Transactions on Signal Processing.55,No.5.2007.[En línea].Recuperado en 2020-2263-72.Disponible en:https://doi.org/10.1109/TSP.2007.892710.

KOST,S.Oracle Critical Patch Updates: Insight and Understanding,En:Integrigy Corporation.25,No.1.2011.[En línea].Recuperado en 2020-1-32.Disponible en: https://www.integrigy.com/files/Integrigy_Oracle_CPU_October_2011_Oracle_Database_Impact%20v1.pdf.

MARK,P.Capability Maturity Model for Software,En: Encyclopedia of Software Engineering (American Cancer Society).15,No.1.2002.[En línea].Recuperado en 2020-392-401.Disponible en: https://doi.org/10.1016/j.infsof.2008.01.010.

MEDINA,Nohem y MESIAS,Wilmer.ESCOGER UNA METODOLOGÍA PARA DESARROLLAR SOFTWARE, DIFÍCIL DECISIÓN.Revista Educación en Ingeniería 10.2015,nro.20.pp.98-109.ISSN 10.26507.

MELLADO, D.; FERNANDEZ, E y PIATTINI, M. Complementando a métrica: proceso de ingeniería de requisitos de seguridad para el desarrollo de sistemas de información seguros, En: Tecnimap Sevilla.12, No.1.2006. [En línea]. Recuperado en 2020-1-12.Disponible en: https://s3.amazonaws.com/academia.edu.documents/57612046/complementando_a_metrica.pdf.

MELLANO, D.; FERNANDEZ,E y PIATTINI,M.A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems, En: Computer Standards & Interfaces.29,No.2.2007.[En línea].Recuperado en 2020-244-53.Disponible en: https://doi.org/10.1016/j.csi.2006.04.002.

MERA, C.Concepto, aplicación y modelo de prospectiva estratégica en la administración de las organizaciones, En: Revista Estrategia Organizacional.1,No.4.2012.[En línea].Recuperado en 2020-25.Disponible en: https://doi.org/10.22490/25392786.1208.

MICROSOFT. [sitio web]. Estados Unidos:SDLC.Microsoft SDL Progress Report.[Consulta: 4 de marzo 2020].Disponible en:https://www.microsoft.com/en-us/download/details.aspx?id=14107

MIN, L y CHEN,T.Analysis of FIPS 140-2 Test and Chaos-Based Pseudorandom Number Generator,En:Revista Chilena de Ingeniería.8,No.1.2013.[En línea].Recuperado en 2020-1-8.Disponible en: http://www.cmsim.eu/papers_pdf/april_2013_papers/4_CMSIM_Journal_2013_Min_Chen_Zang_2_273-280.pdf.

MINCIENCIAS. [sitio web]. Bogotá:CEO,Ministerio de Ciencia Tecnología e Innovación.[Consulta: 26 de febrero 2020].Disponible en:https://minciencias.gov.co/.

MIRNA, M.; GASCA, G y VALTIERRA,C.Caracterizando las Necesidades de las Pymes para Implementar Mejoras de Procesos Software: Una Comparativa entre la Teoría y la Realidad, En:RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação.1,No.15.2014.[En línea].Recuperado en 2020-1-15.Disponible en: https://doi.org/10.4304/risti.e1.1-15.

MITRA, R.Security Level Identification and Secure Software Design of Safety Critical Embedded Systems: Methodologies and Process, En: Incose International Symposium.27,No.1.2017.[En línea].Recuperado en 2020-1300-1313.Disponible en:https://doi.org/10.1002/j.2334-5837.2017.00429.x.

MOHAMMAD, A.; ALQATAWNA, J y MOHAMMAD,A.Secure software engineering: Evaluation of emerging trends, En: 8th International Conference on Information Technology (ICIT).8,No.11.2017.[En línea].Recuperado en 2017-814-18.Disponible en: https://doi.org/10.1109/ICITECH.2017.8079952.

MWANGI,E,MASUPE,S,Y GASENNELWE,M.Formal Specification for Internet of Things Malware,en 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE) (2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)[En línea].2018,Vol.144Nro.49.[Consultado 25 de marzo 2020].Disponible en: https://doi.org/10.1109/iCCECOME.2018.8659285.

MYUNG, S.A Design of Configuration Management Practices and CMPET in Common Criteria Based on Software Process Improvement Activity, En: Computational Science and Its Applications ICCSA.2, No.3.2004.[En línea].Recuperado en 2020-481-90.Disponible en: https://doi.org/10.1007/978-3-540-24707-4_59.

National Security and Competitiveness - ProQuest. [sitio web].Bogotá:IDC,A Study on How Cyber Economic Espionage Affects .[Consulta: 22 de marzo 2020].Disponible en:https://search.proquest.com/openview/7554ba20146515e188f6a371c1e1bea5/1?pq-origsite=gscholar&cbl=18750&diss=y.

NGUYEN, Q.Concepto, Non-functional requirements analysis modeling for software product lines, En:ICSE Workshop on Modeling in Software Engineering.1,No.4.2009.[En línea].Recuperado en 2020-56-61.Disponible en: https://doi.org/10.1109/MISE.2009.5069898.

NIVIA,R,CORTES,P,Y ROJAS,E.Implementation Phase Methodology for the Development of Safe Code in the Information Systems of the Ministry of Housing ,en

Computational Science and Its Applications – ICCSA 2018, ed. Osvaldo Gervasi et al [En línea].2018,Vol.74Nro.1.[Consultado 25 de marzo 2020].Disponible en: https://doi.org/10.1007/978-3-319-95165-2_3.

NOOPUR, D y MULLANEY,J.The Team Software Process (TSP) in Practice: A Summary of Recent Results.IEEE Software [En línea].2003,Vol.2Nro.1.[Consultado 11 de noviembre 2019].Disponible en:https://doi.org/10.1184/R1/6585302.v1.

POTTER, B.A supporting tool for creating and maintaining security targets according to ISO/IEC 15408, En: IEEE International Conference on Computer Science and Automation Engineering.3, No.3.2012. [En línea]. Recuperado en 2020-745-49.Disponible en: https://doi.org/10.1109/ICSESS.2012.6269574.

POTTER. B.Microsoft SDL Threat Modelling.network Security [En línea].2009,Vol.2009Nro.1.[Consultado 27 de marzo 2020].Disponible en: https://sci-hub.tw/10.1016/s1353-4858(09)70008-x.

POTTI, O .;ILLIASHENKO,O y KOMIN,D.Advanced Security Assurance Case Based on ISO/IEC 15408, En: Theory and Engineering of Complex Systems and Dependability, ed. Wojciech Zamojski et al.365,No.11.2015.[En línea].Recuperado en 2020-391-401.Disponible en: https://doi.org/10.1007/978-3-319-19216-1_37.

RAMACHANDRAN, M.Software Security Requirements Management as an Emerging Cloud Computing Service, En: International Journal of Information Management.36,No.4.2016.[En línea].Recuperado en 2020-580-90.Disponible en: https://doi.org/10.1016/j.ijinfomgt.2016.03.008.
RAMOS, B.Avances en criptología y seguridad de la información, En: Ediciones Diaz Santos.1,No.1.204.[En línea].Recuperado en 2020-15-59.Disponible en: https://books.google.com.co.

RODRIGUEZ,G,.ES SEGURO EL USO DEL SOFTWARE EN EL INTERCAMBIO DE INFORMACION BANCARIA.IEEE Software [En línea].2018,Vol.19Nro.1.[Consultado 28 de diciembre 2019].Disponible en:https://repository.unad.edu.co/bitstream/handle/10596/23770/52032204.pdf?sequence=5&isAllowed=y.

ROMERO, A.; MIRANDA, A y MIRANDA,S.Indicadores de un Sistema de Gestión y sus errores, En:Gestionpolis.,No.3.2013.[En línea].Recuperado en 2020-14-103.Disponible en: https://doi.org/10.4067/S0718-07642013000300012.

RUSSO, J y SOLARI, M. Estudio de Mapeo Sistemático sobre Arquitecturas de Software para Big Data, En: Researchdate.19, No.3.2017. [En línea]. Recuperado en 2017-10-357.Disponible en: https://www.researchgate.net/publication/331001020_Estudio_de_Mapeo_Sistematico_sobre_Arquitecturas_de_Software_para_Big_Data.

SALO,O,ABRAHAMSSON,P.Agile Methods in European Embedded Software Development Organisations: A Survey on the Actual Use and Usefulness of Extreme Programming and Scrum. IET Software [En línea].2008, Vol.2Nro.1. [Consultado 20 de diciembre 2019]. Disponible en:https://doi.org/10.1049/iet-sen:20070038.

SEUNG KOU, K.;JAE GOO,J y GANG SOO,L.Definition of Evaluation Assurance Levels and Estimation of Evaluation Efforts for Operational System Based ISO/IEC 19791, En:International Conference on Security Technology.1,No.1.2008.[En línea].Recuperado en 2020-1-12.Disponible en: https://doi.org/10.1109/SecTech.2008.41.

SGUERRA,David. Diseño de software seguro.4 ed.Bogotá D.C:2006.8p.(acistente;nro 43).ISBN-41-0131-5.

SHANDI,A,LAWNEH,A,Y JARADAT,R.Cloud Security Engineering: Early Stages of SDLC, Future Generation Computer Systems [En línea].2017,Vol.74Nro.1.[Consultado 25 de marzo 2020].Disponible en: https://doi.org/10.1016/j.future.2016.10.005.

SILVA, P.; , MATALONGA,S.;NOEL,R y ASTUDILLO,H .Methodologies to Identify and Mitigate Security Threats in Software Development: Two Systematic Mapping Studies, Researchdate.19,No.3.2016.[En línea].Recuperado en 2016-10-19153.Disponible en:https://www.researchgate.net/publication/311424013_Methodologies_to_Identify_and_Mitigate_Security_Threats_in_Software_Development_Two_Systematic_Mapping_Studies.

SLUPSK,J,TADDEO,M.Generative Metaphors in Cybersecurity Governance, en The Yearbook of the Digital Ethics Lab, ed. Christopher Burr y Silvia Milano, Digital Ethics Lab Yearbook [En línea].2020,Vol11.Nro.30.[Consultado 26 de marzo 2020].Disponible en:https://doi.org/10.1007/978-3-030-29145-7_2.

SOBEL, A y MCGRAW,G.Interview: Software Security in the Real World, En: Computer.43,No.9.2010.[En línea].Recuperado en 2020-47-53.Disponible en: https://doi.org/10.1109/MC.2010.256.

SPEARING,M.Modification of the Clinical Global Impressions (CGI) Scale for Use in Bipolar Illness (BP) The CGI-BP,En: Psychiatry Research.73,No.3.1997.[En línea].Recuperado en 2020-71-159.Disponible en:https://doi.org/10.1016/S0165-1781(97)00123-6.

STRAKER, K.; WRIGLEY, C y ROSEMANN,M.Typologies and Touchpoints: Designing Multi-Channel Digital Strategies, En:Journal of Research in Interactive Marketing.9,No.2.2015.[En línea].Recuperado en 2020-110-28.Disponible en: https://doi.org/10.1108/JRIM-06-2014-0039.

SUKHAI,N.Hacking and cybercrime.Kennesaw, Georgia: Association for Computing Machinery [En línea].2004,Vol.19Nro.1.[Consultado 11 de noviembre 2019].Disponible en: https://doi.org/10.1145/1059524.1059553.

SURAKHI, O, A Survey on Design Methods for Secure Software Development. International Journal of computers & technology. [En línea].2017, Vol.16Nro.7. [Consultado 22 de marzo 2020]. Disponible en:https://doi.org/10.24297/ijct.v16i7.6467.

TALLON, J.Common Criteria: Herramienta para el desarrollo seguro, En: Jtsec Beyond it Security.3,No.1.2018.[En línea].Recuperado en 2020-10-59.Disponible en: https://es.slideshare.net/javitallon/common-criteria-herramienta-para-el-desarrollo-seguro-97058338.

TONDEL, I.Understanding challenges to adoption of the Microsoft elevation of privilege game, En: Understanding challenges to adoption of the Microsoft elevation of privilege game.18,No.11.2018.[En línea].Recuperado en 2020-1-10.Disponible en: https://doi.org/10.1145/3190619.3190633.

VELASCO, C,Y HARDANY,E.El mundo cibernético y los delitos informáticos. Repositorio Institucional USC [En línea].2019,Vol.64Nro.1.[Consultado 22 de marzo 2020].Disponible en:https://repository.usc.edu.co/handle/20.500.12421/510.
VON SOLMS, B.Information Security Governance: COBIT or ISO 17799 or Both, En:Computers & Security.24,No.2.2005.[En línea].Recuperado en 2020-99-104.Disponible en: https://doi.org/10.1016/j.cose.2005.02.002.

WANG,F.The Best-of-2-Worlds Philosophy: Developing Local Dismantling and Global Infrastructure Network for Sustainable e-Waste Treatment in Emerging Economies,En:Waste Management, Special Thematic Issue: Waste Management in Developing Countries.32,No.11.2012.[En línea].Recuperado en 2020-46-2134.Disponible en:https://doi.org/10.1016/j.wasman.2012.03.029.

WELICKI,L.Patrones y Antipatrones: una Introducción - Parte II,En:microsoft.1,No.1.2008.[En línea].Recuperado en 2020-12-50.Disponible en: https://docs.microsoft.com/es-es/previous-versions/bb972251(v%3dmsdn.10)

WICHERS, D. OWASP Top-10 2017, En: Owasp the Open Web Application Securty Project .1, No.10.2017. [En línea]. Recuperado en 2017 -1-43. Disponible en: https://upload.wikimedia.org/wikipedia/mediawiki/archive/e/e9/20180111214627%21OWASP_Top-10_2017_-_Presentation.pdf.

ZIMMERMANN, J.OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection: IEEE Transactions on Communications.28, No.4.1980.

[En línea]. Recuperado en 2020-32-425.Disponible en: https://doi.org/10.1109/TCOM.1980.1094702

ANEXOS

ANEXO A Resumen Analítico Especializado RAE

Fecha de Realización:	05/09/2020
Programa:	Especialización en seguridad Informática
Línea de Investigación:	Gestión de sistema
Título:	Análisis comparativo entre metodologías para el desarrollo software seguro de acuerdo con el estándar ISO/IEC 15408
Autor(es):	Bayona Guio Nancy Soraida
Palabras Claves:	Desarrollo, seguro, metodologías, estándares, fases
Descripción:	En este trabajo, se realizó un estudio con siete metodologías para el desarrollo seguro debido que se identificó que una gran parte de los desarrollares no conocen de metodologías de desarrollo seguro, ni mucho menos aplican de buenas prácticas de seguridad a través de esta monografía, se realizó un análisis comparativo entre las metodologías que fueron objeto de estudio, donde se identificaron unos criterios con base al estándar internacional ISO/IEC 15408 de los cuales fueron vitales para realizar este análisis, con el fin de identificar cuáles de estas monografías cumplía con la mayoría de criterios definidos por el estándar, también se realizó un análisis con base al tamaño de los grupos de desarrollo con el propósito de identificar con cada una de las metodologías cuales de estas se adaptan de acuerdo al número de colaboradores dentro de una organización y finalmente generar un serie de recomendaciones mínimas que todo desarrollador debe tener en cuenta al momento codificar.
Fuentes bibliográficas destacadas: ALMEIDA,G.Legal Rules and Information Security Technical Standards: Possible Approach for Filling in the Blanks of Cybercrime Legislation. SSRN Electronic Journal [En línea].2011,Vol.2Nro.1.[Consultado 20 de febrero de 2020].Disponible en: https://doi.org/10.2139/ssrn.1742962.	

BRITO, Metodologías para desarrollar software seguro, En: Universidad Autónoma de Zacatecas.2, No.3.2013. [En línea]. Recuperado en 2020-2-18. Disponible en: https://www.redalyc.org/pdf/5122/512251564005.pdf.
BROWN, A.Realizing service-oriented solutions with the IBM Rational Software Development Platform, En:IBM.44,No.4.2005.[En línea].Recuperado en 2020-727-52.Disponible en: https://doi.org/10.1147/sj.444.0727.
CADAVID, A. Revisión de metodologías ágiles para el desarrollo de software.Prospectiva [En línea].2013,Vol.11Nro.2.[Consultado 20 de noviembre de 2019].Disponible en:https://doi.org/10.15665/rp.v11i2.36.

CARDOZO, D.; VELASQUEZ y RODRIGUEZ, Revisión de la definición de PYME en América Latina, En:10th Latin American and Caribbean Conference for Engineering and Technology. Megaprojects.10, No.1.2012. [En línea]. Recuperado en 2020-1-10. Disponible en: http://oa.upm.es/19446/
FUTCHER,L y VON SOLMS,R.Guidelines for secure software development, En: Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries.8,No.1.2008.[En línea].Recuperado en 2020-56-65.Disponible en: https://doi.org/10.1145/1456659.1456667.

MELLANO, D.; FERNANDEZ,E y PIATTINI,M.A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems, En: Computer Standards & Interfaces.29,No.2.2007.[En línea].Recuperado en 2020-244-53.Disponible en: https://doi.org/10.1016/j.csi.2006.04.002.

WELICKI,L.Patrones y Antipatrones: una Introducción - Parte II,En:microsoft.1,No.1.2008.[En línea].Recuperado en 2020-12-50.Disponible en: https://docs.microsoft.com/es-es/previous-versions/bb972251(v%3dmsdn.10)

ZIMMERMANN, J.OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection: IEEE Transactions on Communications.28, No.4.1980. [En línea]. Recuperado en 2020-32-425.Disponible en: https://doi.org/10.1109/TCOM.1980.1094702

Contenido del documento:	Investigaciones ha demostrado en los últimos tiempos que la gran mayoría de las vulnerabilidades dentro de los sistemas de información se ejecuta a través del código, debido que se presenta una debilidad dentro de los equipos de ingeniería, donde no se aplica buenas prácticas de desarrollo seguro, dado que para muchos desarrolladores existe

	un desconocimiento sobre, la existencia de metodologías para el desarrollo de software seguro, de ahí surge la necesidad de indagar que metodologías de desarrollo seguro existe, como se podrían implementar y aplicar dentro de los equipos de desarrollo de acuerdo al número de integrantes; por consiguiente, se realizó un análisis comparativo, entre siete metodologías de desarrollo de software seguro, bajo un estándar internacional Common Criteria (ISO/IEC 15408). Este estudio se realizó a través de unas fases y actividades, divididas en tres capítulos, que permitieron describir conceptos, antecedentes, características y finalmente realizar un análisis comparativo entre las metodologías de desarrollo de software seguro y como objeto de estudio, se observa que la metodología de desarrollo seguro, que cumplió con la mayoría de los criterios establecidos, dentro del estándar internacional es la metodología Security Development Lifecycle SDL. La investigación permite concluir que hay criterios que hasta el momento ninguna de las metodologías los tiene presentes dentro sus fases, por consiguiente, se considera que estos aspectos se puedan tener en cuenta para futuras investigaciones.
Marco Metodológico:	No aplica
Conceptos adquiridos:	Con el desarrollo de este trabajo se adquirió conocimiento al saber que existen metodologías que se, pueden aplicar dentro de un ciclo de vida de desarrollo. Se adquirió conocimientos en aprender técnicas para proteger el código al momento de desarrollar. Se logro identificar la importancia de implementar metodologías para el desarrollo seguro.

Conclusiones:	Gracias al análisis comparativo se logró concluir, de las siete metodologías seleccionadas para el desarrollo de software seguro, la metodología que cumplió con la mayoría de los criterios de acuerdo con el estándar internacional Common Criteria (ISO/IEC 15408) es; Security Development LifeCycle (SDLC), dado que en todas las fases del SDLC se realiza una evaluación de estado, es decir desde la fase de requisitos hasta la liberación, a través de un proceso de seguimiento. Con base en el estándar Common Criteria (ISO/IEC 15408), se logró definir los criterios requeridos con los cuales se realizó un análisis comparativo entre las metodologías de desarrollo seguro, con el fin de identificar cuáles de estas cumplen con requerimientos exigidos por el estándar. Con la elaboración de una manual de desarrollo seguro se brinda un apoyo a los desarrolladores, debido que la gran mayoría de ellos no conocen de metodologías para el desarrollo seguro, ni mucho menos, su aplicación dentro de sus procesos de desarrollo.