UNIVERSIDAD DE EXTREMADURA

La Seguridad en el Desarrollo de Software implementada de forma Preventiva

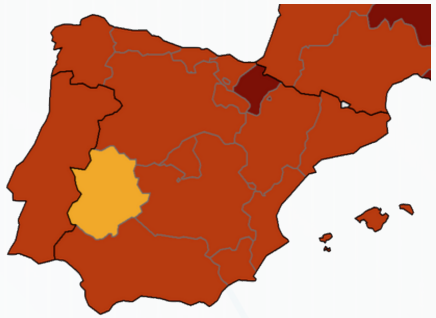
José Carlos Sancho Núñez
(jcsanchon@unex.es)

Profesor Universidad de Extremadura e
Investigador Cátedra Viewnext-UEx

02/12/2021

PRESENTACIÓN

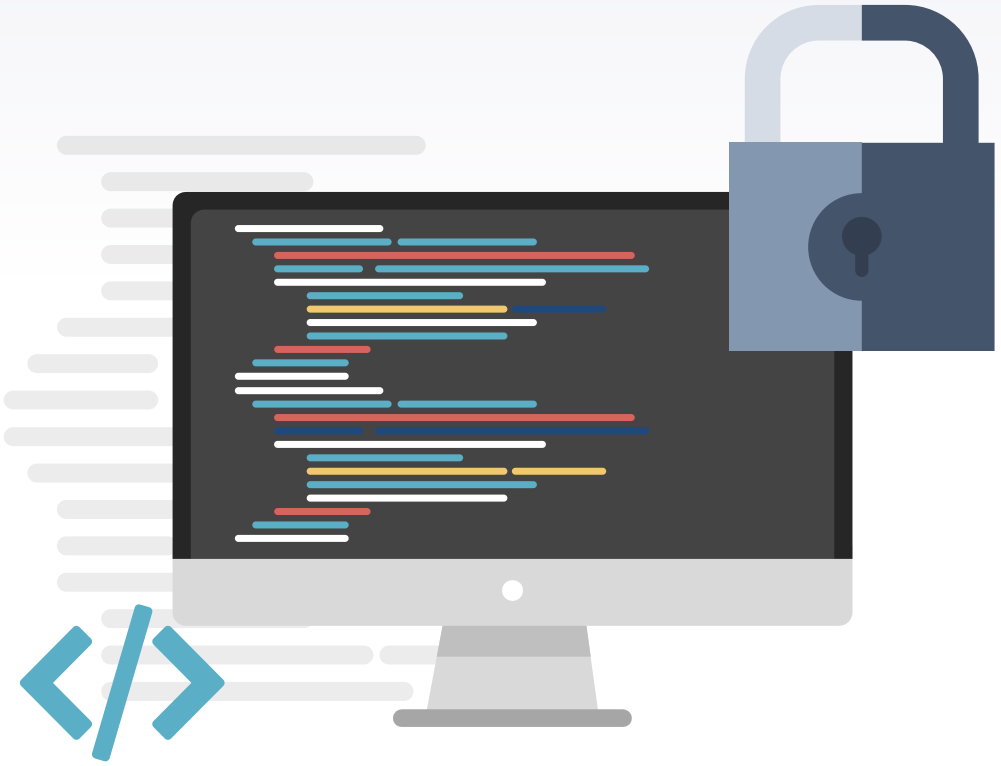
José Carlos
Sancho Núñez



- Ingeniero en Informática
- Doctor en Tecnologías Informáticas
- Profesor Universidad de Extremadura
- Investigador Cátedra Viewnext-UEx
- Miembro del Cuerpo Oficial de Peritos del Colegio Profesional de Ingenieros en Informática de Extremadura (Col. 103)

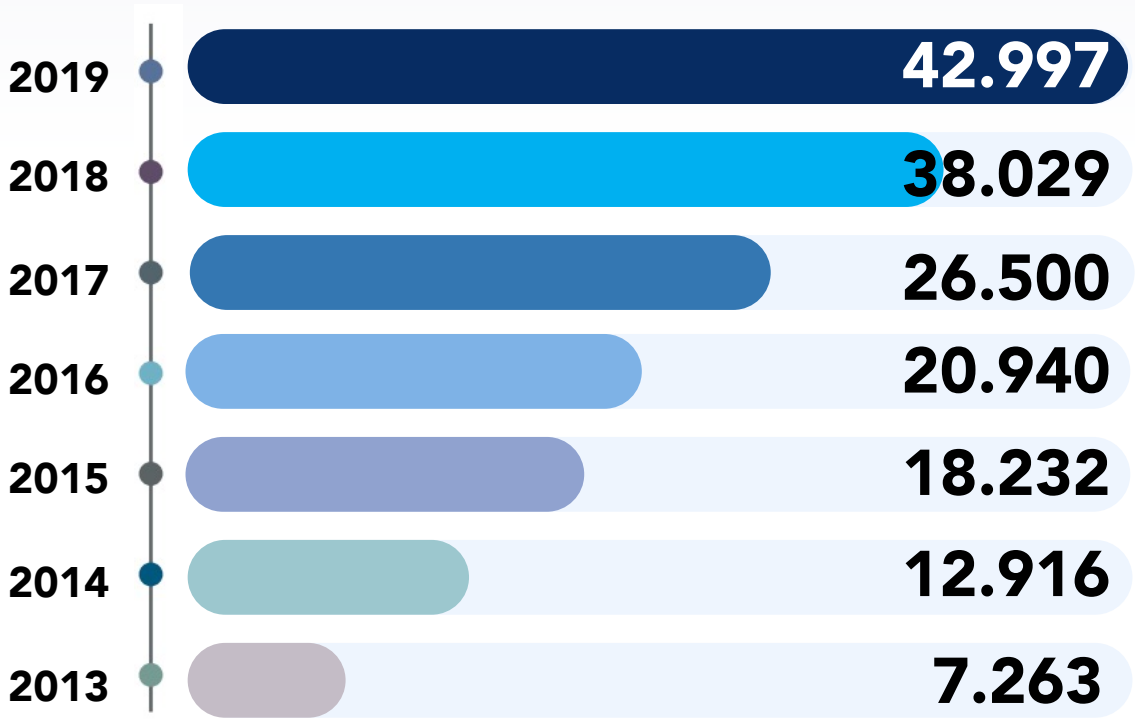
ÍNDICE

Introducción y contexto



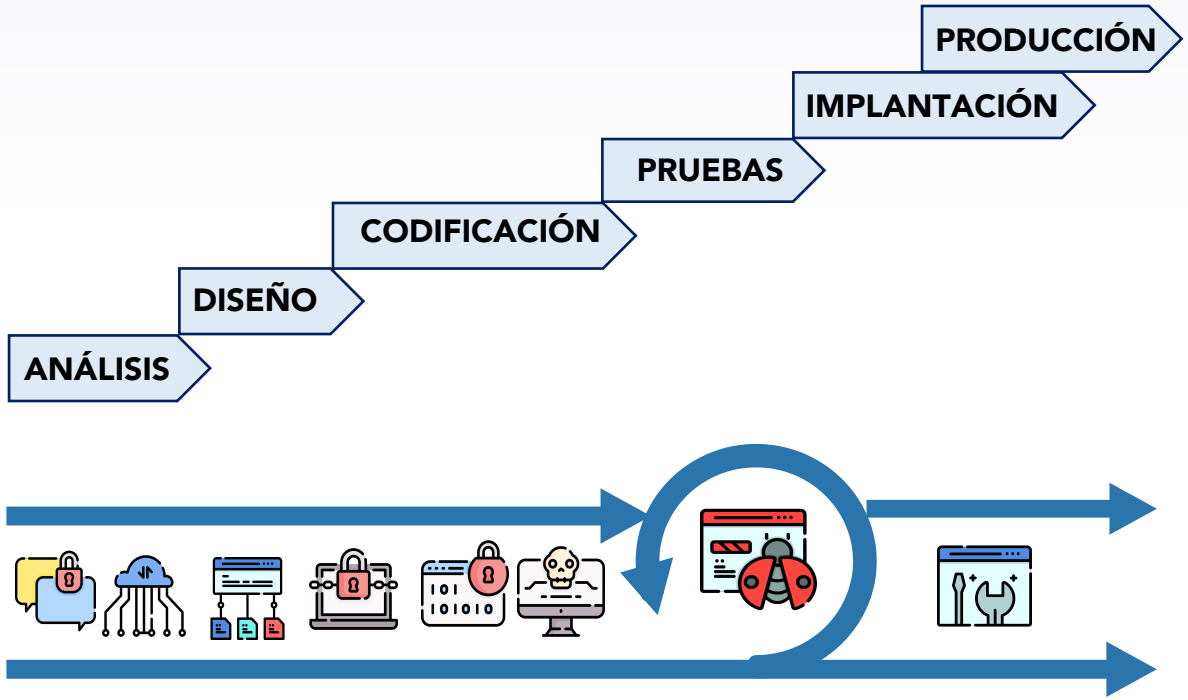
Motivación de la investigación

Evolución de los incidentes gestionados por el CCN-CERT.
Fuente: CCN-CERT.



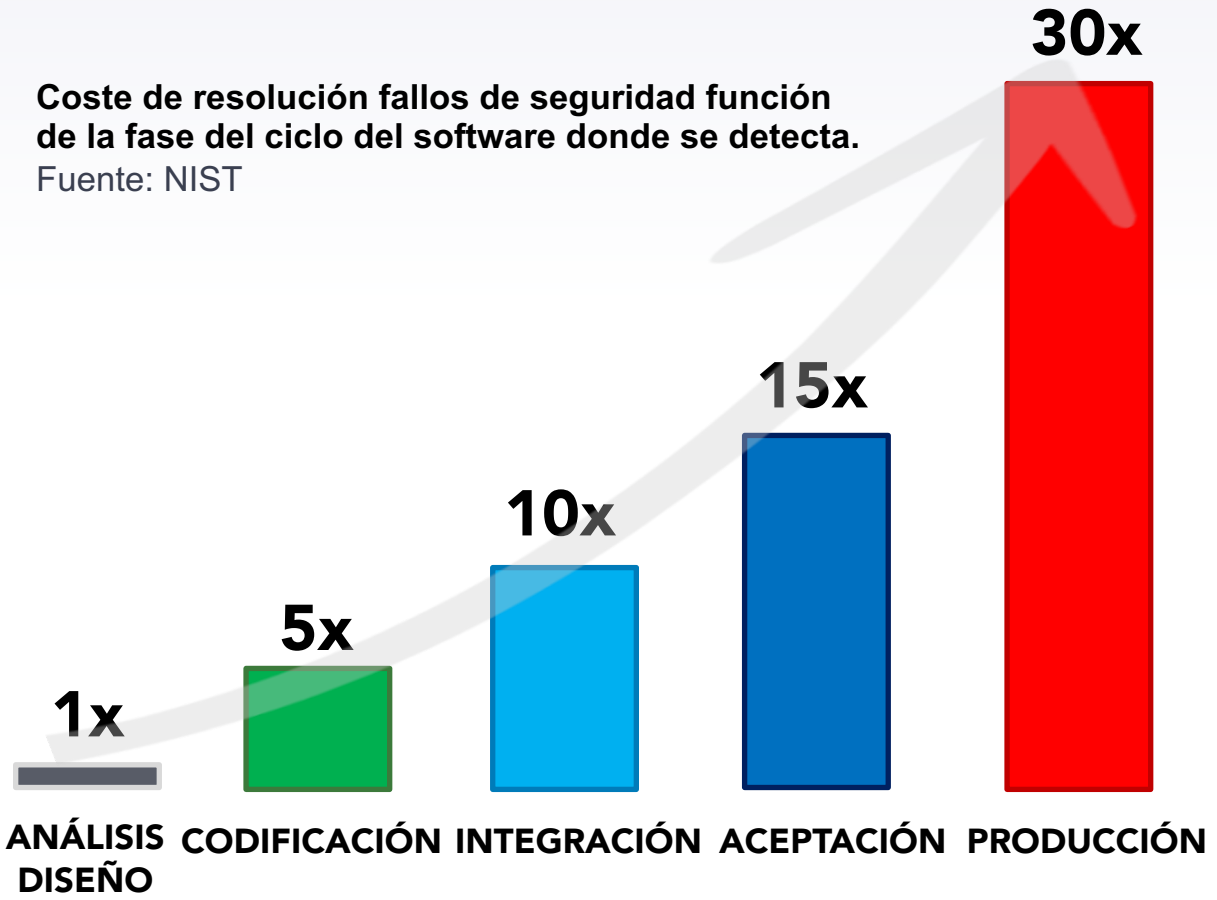
Monitorización de ataques cibernéticos en tiempo real.
Fragmento tomado el 19 de febrero de 2021.
Fuente: Kaspersky.

Motivación de la investigación

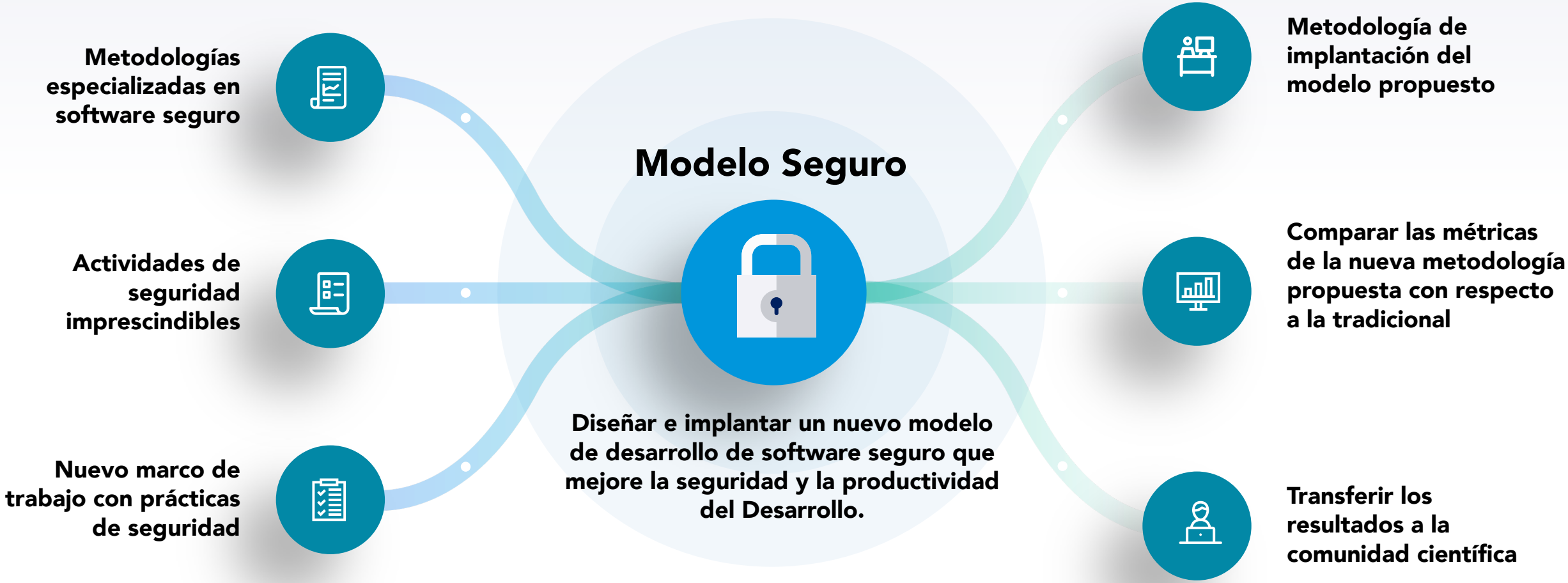


CICLO DE VIDA DEL SOFTWARE

Coste de resolución fallos de seguridad función de la fase del ciclo del software donde se detecta.
Fuente: NIST



Objetivos de la investigación



Seguridad en el ciclo de vida del software

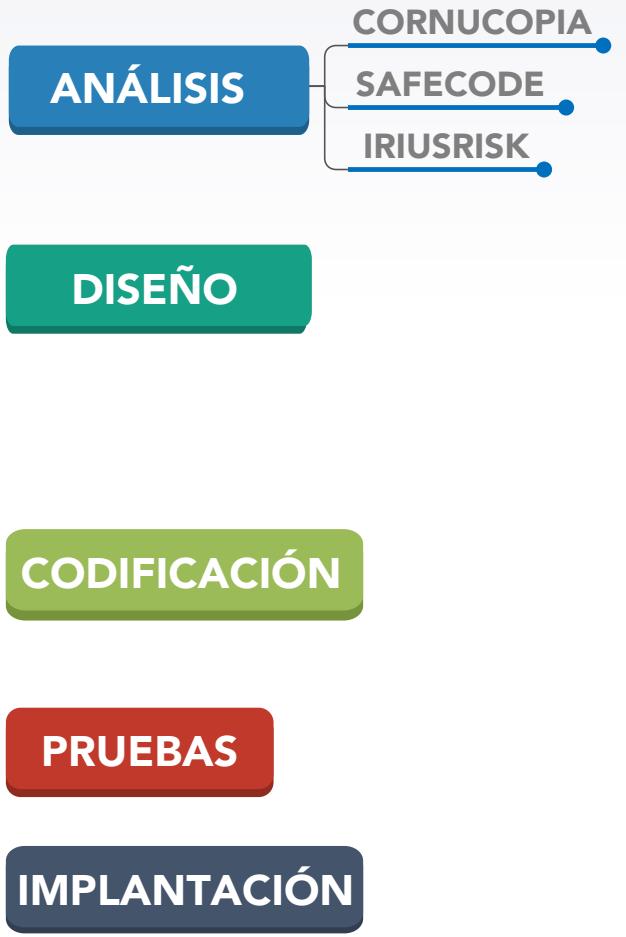
Seguridad análisis – Requisitos de seguridad

Captchas

Número de intentos fallidos

Doble factor autenticación

EVITAR ATAQUES AUTOMATIZADOS



Seguridad en el ciclo de vida del software

Seguridad diseño – Patrones de diseño

Principio menor privilegio

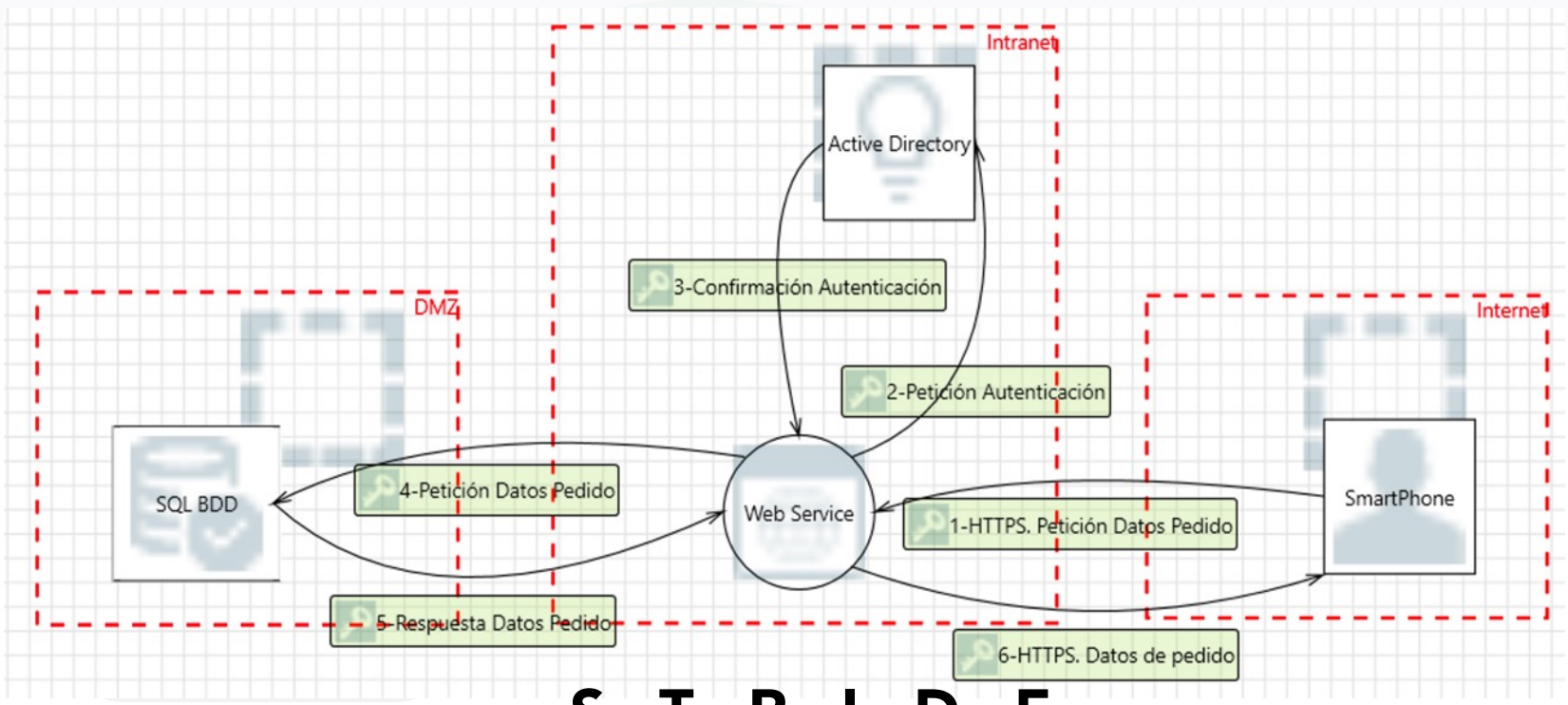
Reducción de ataque

Separación de privilegios

"Fallo Seguro"

Seguridad en el ciclo de vida del software

Seguridad diseño – Modelado de amenazas

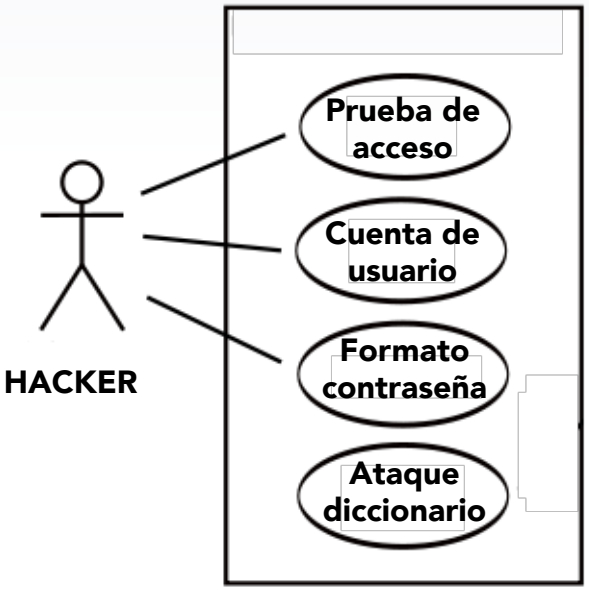


S T R I D E

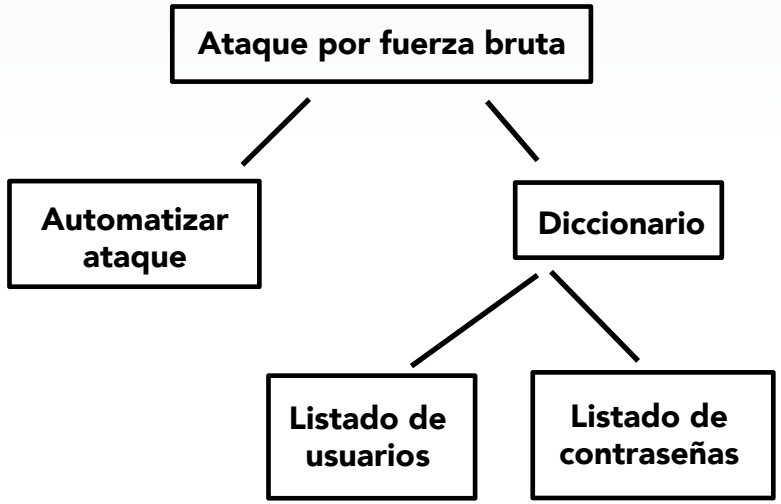
- ANÁLISIS
  - CORNUCOPIA
  - SAFECODE
  - IRIUSRISK
- DISEÑO
  - PATRONES DE DISEÑO
  - MODELADO
- CODIFICACIÓN
- PRUEBAS
- IMPLANTACIÓN

Seguridad en el ciclo de vida del software

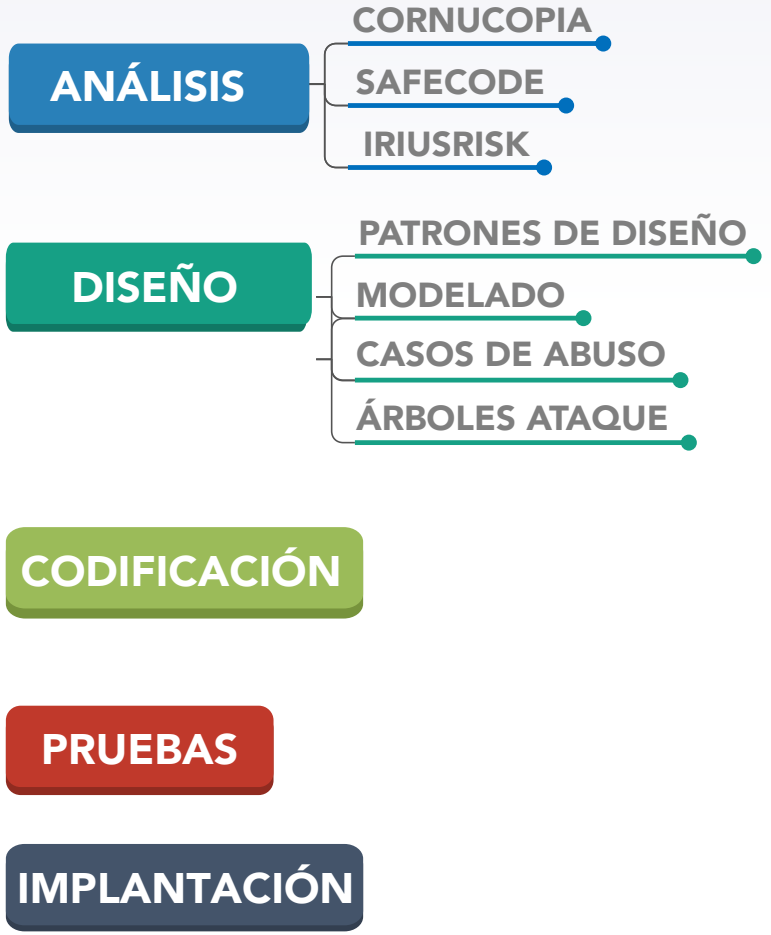
Seguridad diseño – Otras técnicas



Casos de abuso

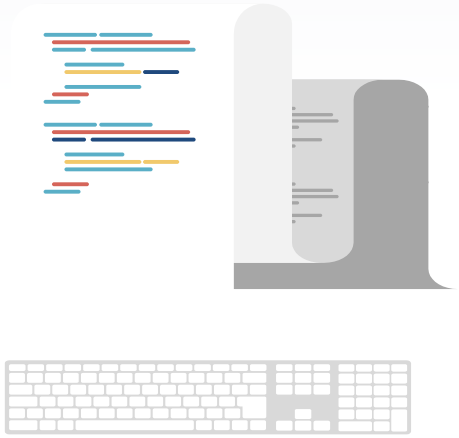


Árboles de ataque



Seguridad en el ciclo de vida del software

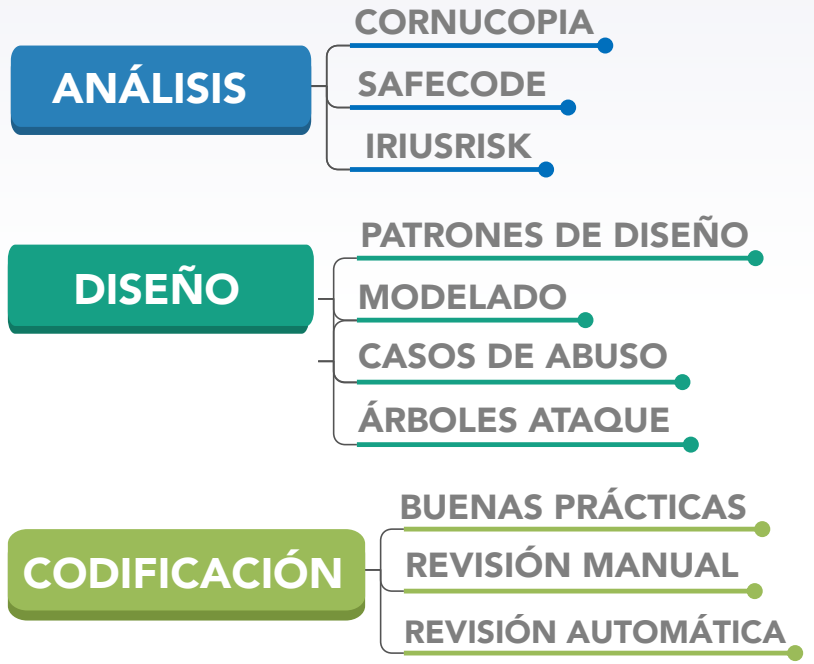
Seguridad en la codificación



Buenas prácticas de codificación

Revisión de código manual

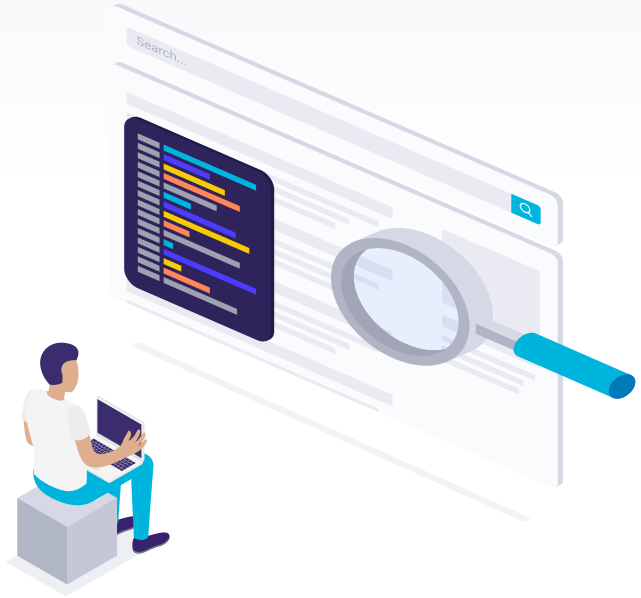
Revisión de código automática



Seguridad en el ciclo de vida del software

Seguridad en las pruebas

Test de penetración



Fuzz testing

- ANÁLISIS
  - CORNUCOPIA
  - SAFECODE
  - IRIUSRISK
- DISEÑO
  - PATRONES DE DISEÑO
  - MODELADO
  - CASOS DE ABUSO
  - ÁRBOLES ATAQUE
- CODIFICACIÓN
  - BUENAS PRÁCTICAS
  - REVISIÓN MANUAL
  - REVISIÓN AUTOMÁTICA
- PRUEBAS
  - TEST DE PENETRACIÓN
  - FUZZ TESTING
- IMPLANTACIÓN

Seguridad en el ciclo de vida del software

Seguridad en la implantación

Proceso de implantación y configuración

Plan de contingencias

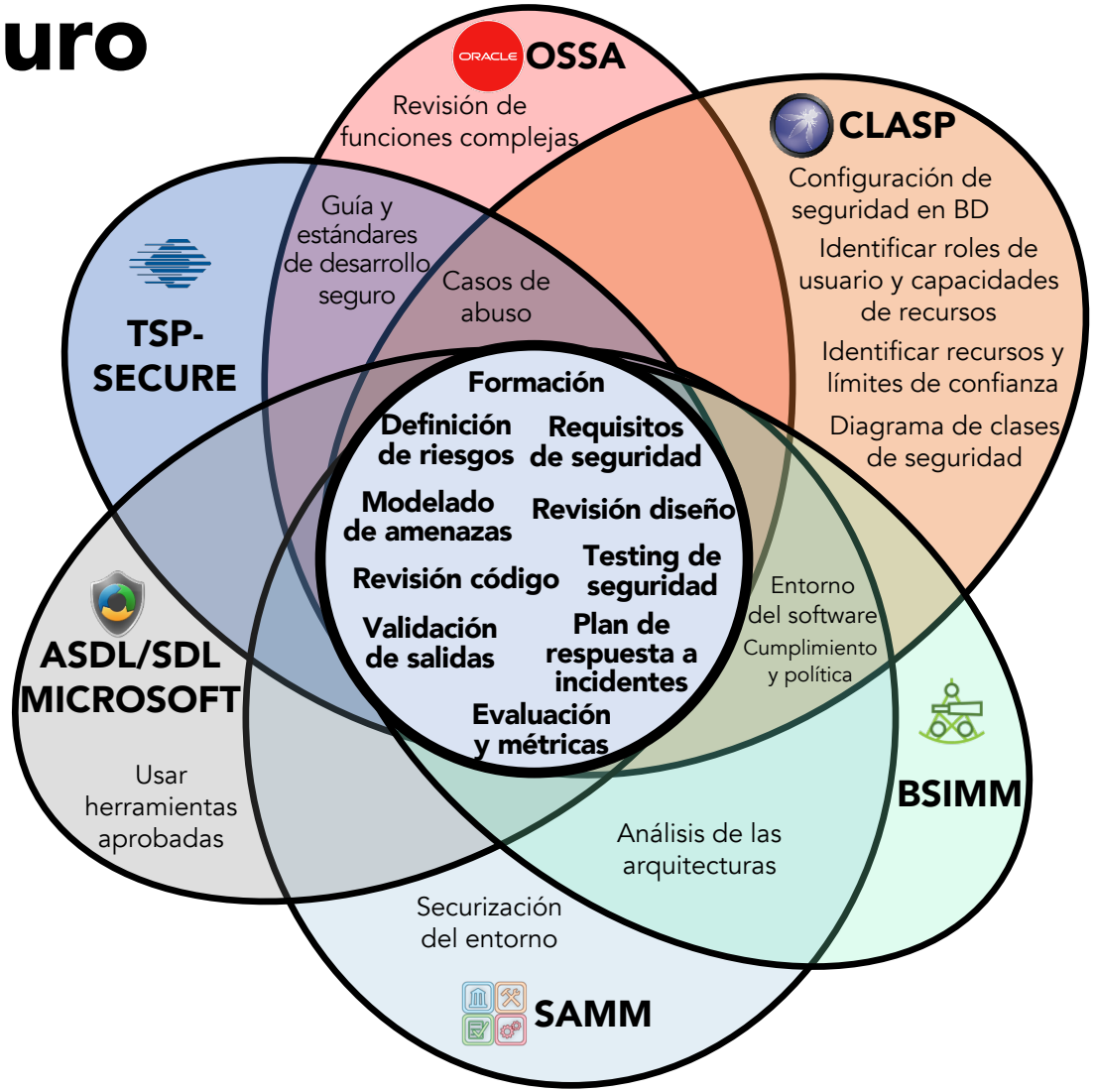
- **ANÁLISIS**
  - CORNUCOPIA
  - SAFECODE
  - IRIUSRISK
- **DISEÑO**
  - PATRONES DE DISEÑO
  - MODELADO
  - CASOS DE ABUSO
  - ÁRBOLES ATAQUE
- **CODIFICACIÓN**
  - BUENAS PRÁCTICAS
  - REVISIÓN MANUAL
  - REVISIÓN AUTOMÁTICA
- **PRUEBAS**
  - TEST DE PENETRACIÓN
  - FUZZ TESTING
- **IMPLANTACIÓN**
  - PLAN DE CONTINGENCIAS

Modelos de Desarrollo Seguro

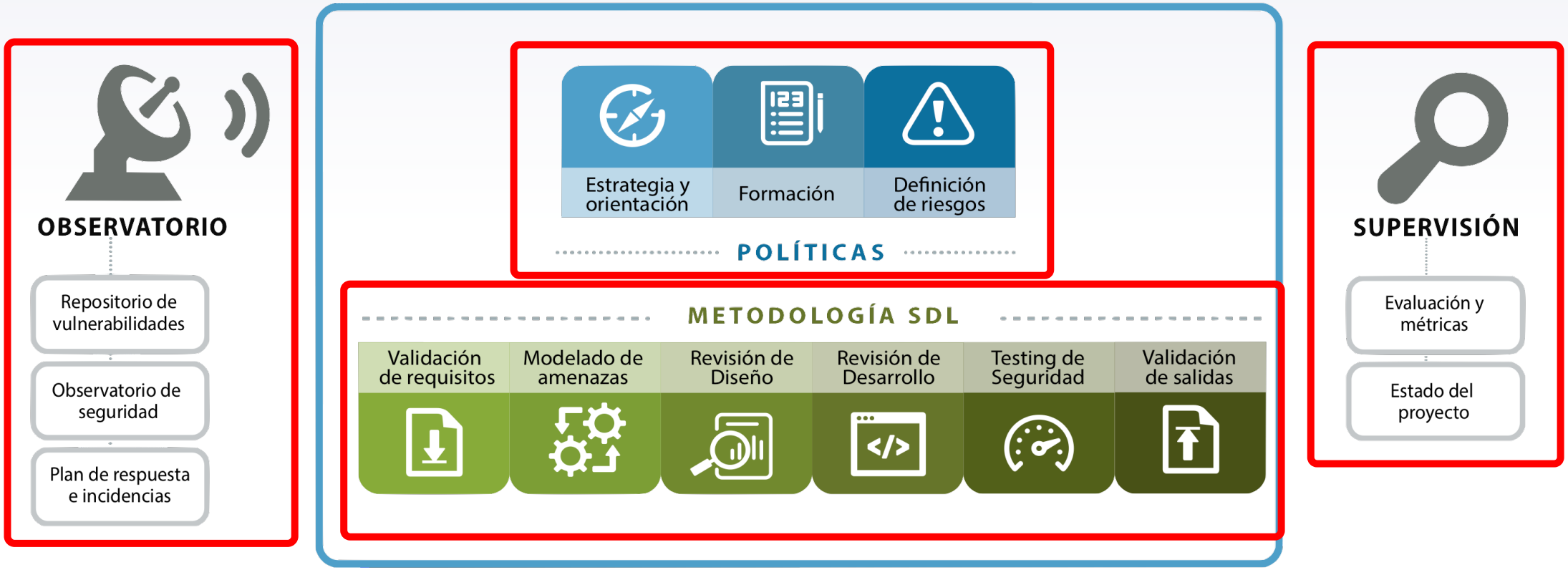
ANÁLISIS DE METODOLOGÍAS DE DESARROLLO SEGURO

Microsoft · Microsoft · OSSA · CLASP · TSP-Secure · SAMM · BSIMM

- **Microsoft SDL**
- **Agile-SDL**
- **Oracle Software Security Assurance**
- **Comprehensive Lightweight Application Security Process**
- **Team Software Process Secure**
- **Software Assurance Maturity Model**
- **Building Security In Maturity Model Framework**

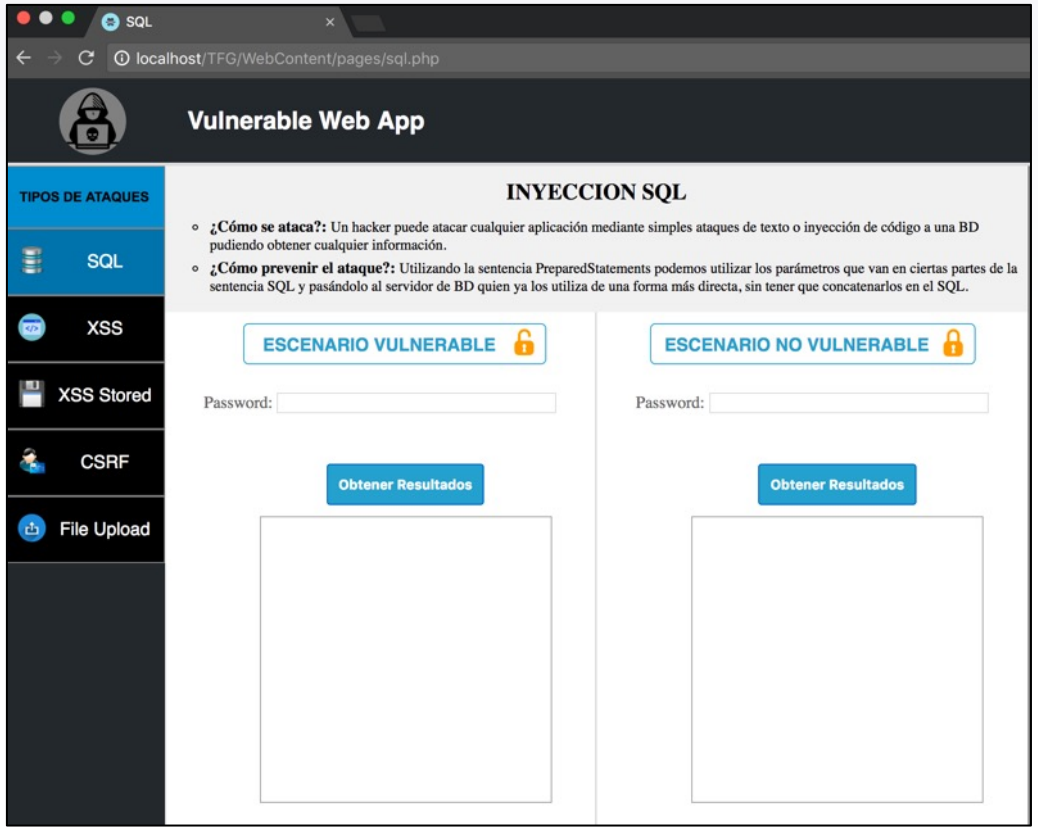


Modelo de Desarrollo Seguro Viewnext-UEx



Preventivo **Integrado** **Flexible** **Retroalimentado** **Reutilizable**

Puntos fuertes



Herramienta entrenamiento seguro

RIESGOS

REQUISITO DE SEGURIDAD

AMENAZAS

BUENAS PRÁCTICAS

@ # ¡!¿?

< > " ' % ( ) & + / \

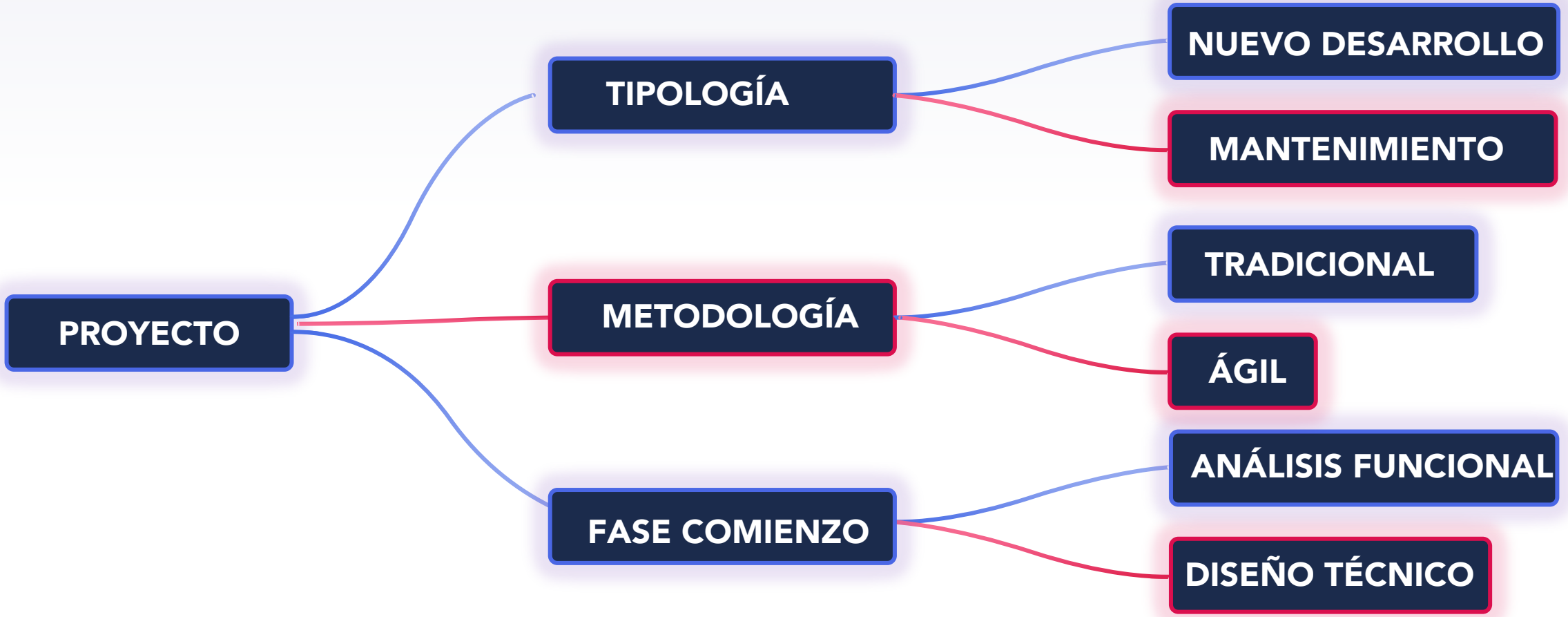
Trazabilidad durante todo el ciclo

Actividades novedosas del Modelo Viewnext-UEx

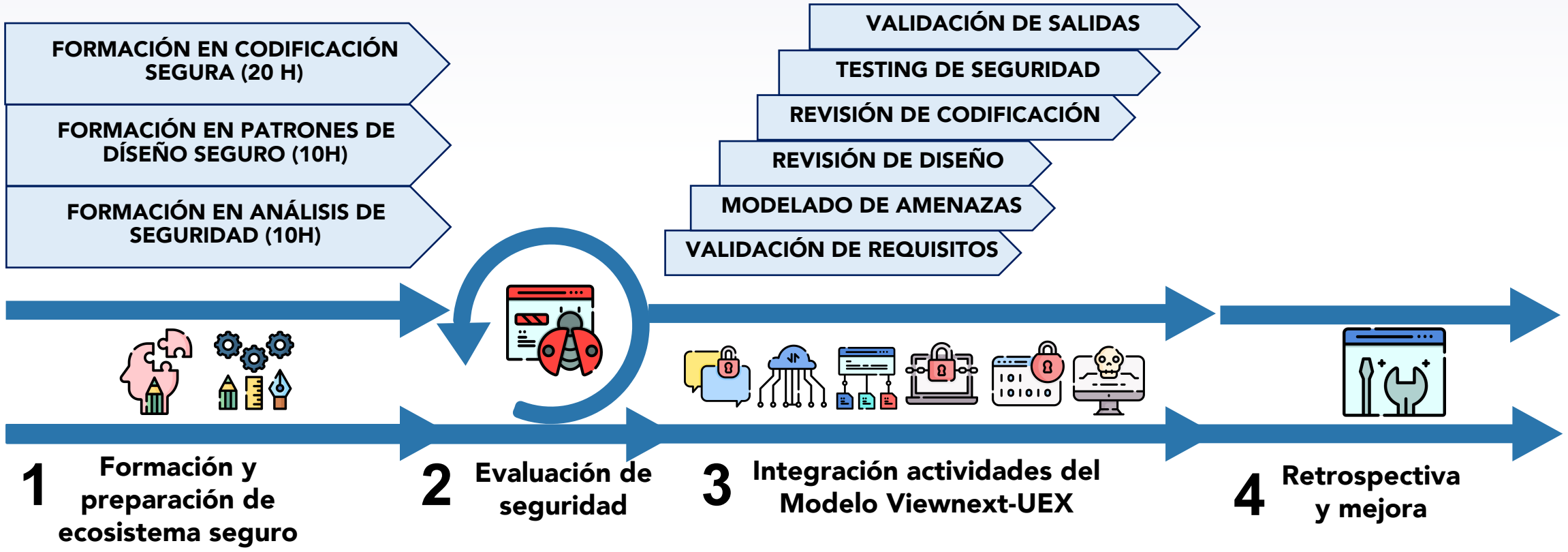
Modelo Viewnext-UEx VS otros

Fase/Modelo	Microsoft SDL/ASDL	OSSA	CLASP	TSP-Secure	SAMM	BSIMM	Viewnext-UEx
Políticas	✗	✗	✓	✗	✓	✓	✓
Formación	✓	✓	✓	✓	✓	✓	✓
Análisis	✓	✓	✓	✓	✓	✓	✓
Diseño	✓	✓	✓	✓	✓	✓	✓
Implementación	✓	✓	✓	✓	✓	✓	✓
Pruebas	✓	✓	✓	✓	✓	✓	✓
Pre-Release	✓	✓	✓	✗	✗	✓	✓
Post-Release	✓	✓	✓	✓	✓	✓	✓
Métricas	✓	✓	✓	✓	✓	✓	✓
Uso empresarial	✓	✓	✓	✓	✓	✓	✓

Características de un proyecto de software



Metodología de implantación



Indicadores: seguridad y productividad

SEGURIDAD

VULNERABILIDAD	CRITICIDAD	CATEGORÍA
Val. entradas	Informativa [0]	Arquitectura
Gestión sesiones	Baja [0.1-3.9]	Desarrollo
Control de Acceso	Media [4.0-6.9]	
Autorización	Alta [7.0-8.9]	
Errores	Crítica [9.0-10]	
Arquitectura		
Configuración		
Protección datos		
Etc.		

PRODUCTIVIDAD

FASE	UNIDAD
Gestión	Horas
Análisis y diseño	
Codificación	
Pruebas	
Evaluación	
Resolución de fallos	
Desarrollo total	

Aplicación experimental: empresa y proyecto

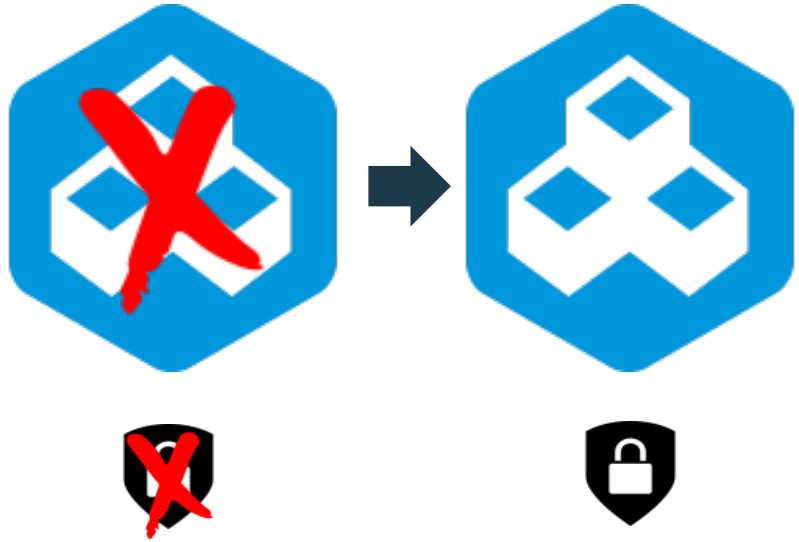
VIEWNEXT AN IBM SUBSIDIARY IBM

Características del proyecto

Caso de estudio

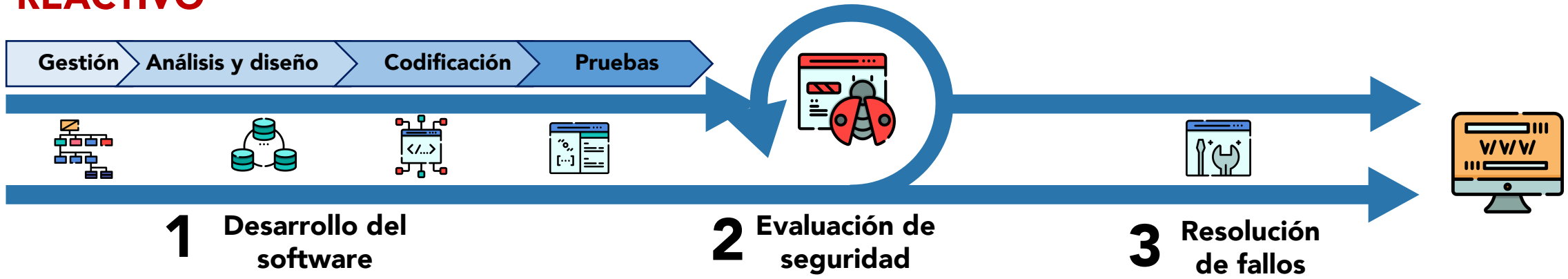
- **Tamaño del Equipo**: +/- 15 desarrolladores
- **Compañía eléctrica**: Criticidad crítica
- **Equipo multifuncional**: Salvo el testing de seguridad
- **Nuevos Desarrollos**: Incluido el análisis
- **Metodología Ágil**: SCRUM – sprint mensual
- **Cliente proactivo**: Interesado en mejorar

MÓDULO 1 → **MÓDULO 2**

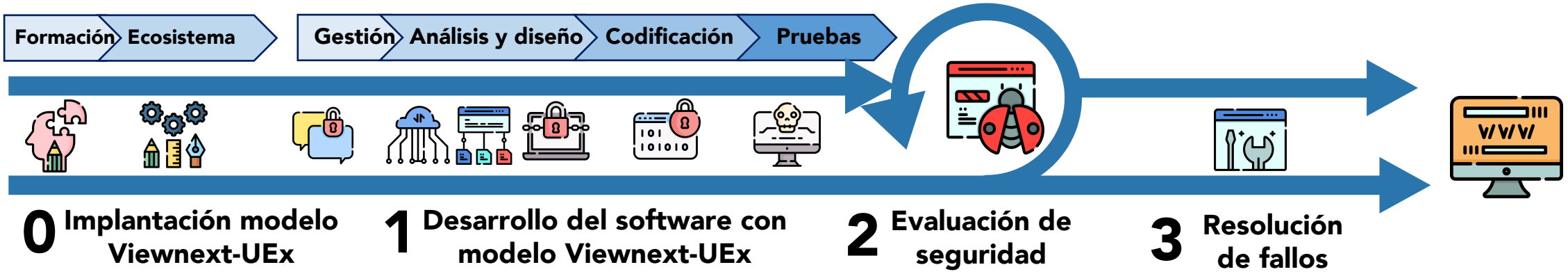


Aplicación experimental: escenarios

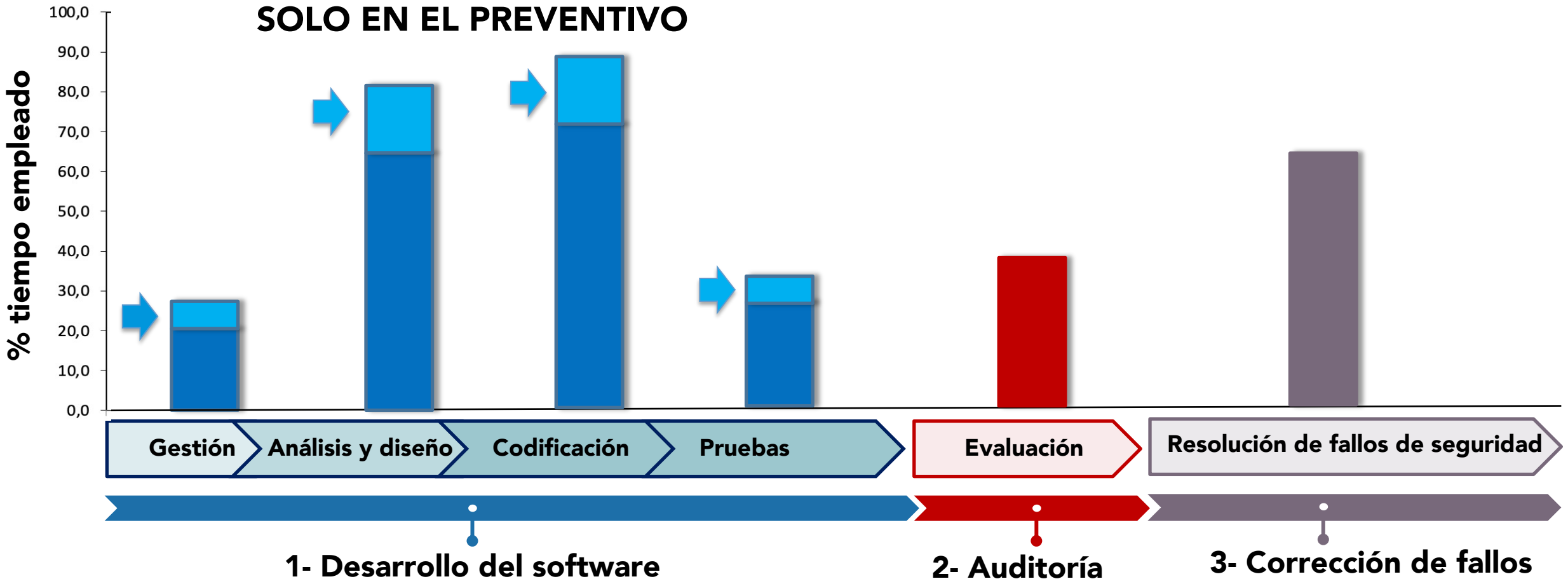
REACTIVO



PREVENTIVO

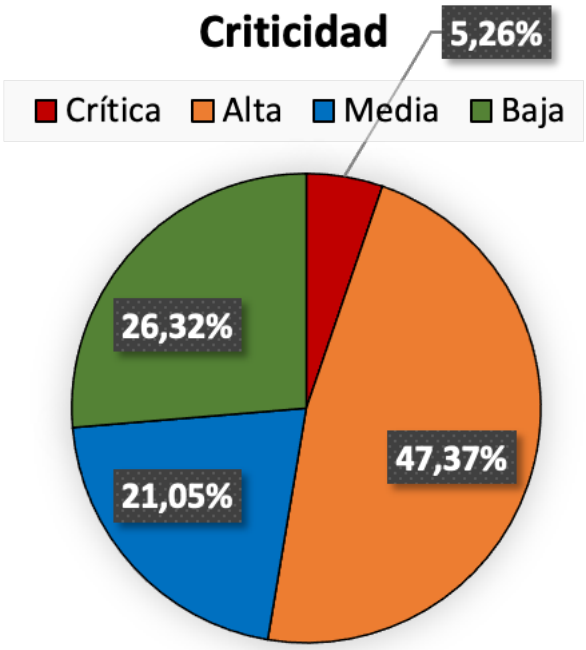


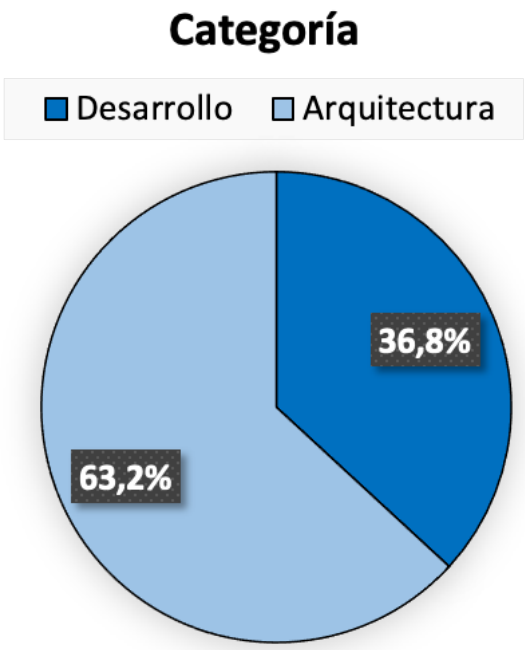
Métricas de productividad

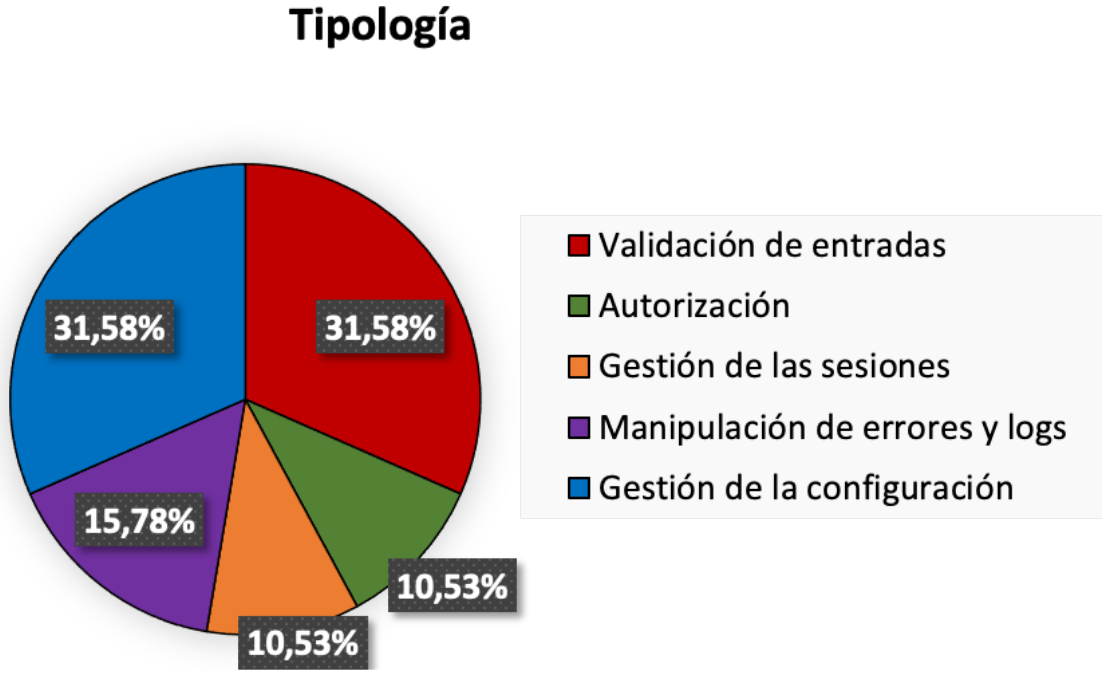


Escenario reactivo: resultados seguridad

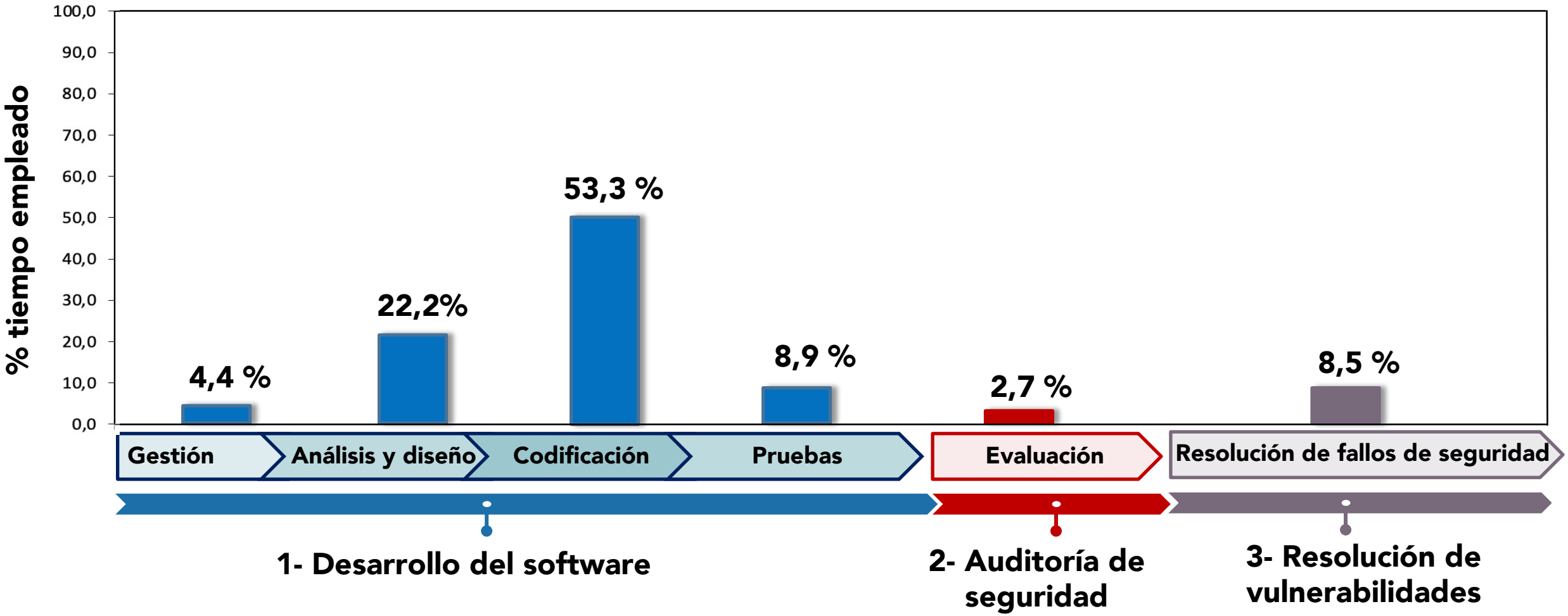
19 VULNERABILIDADES





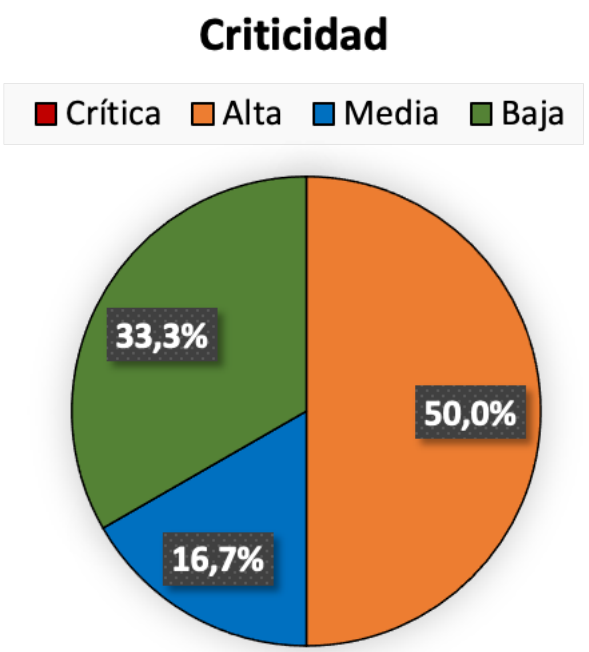


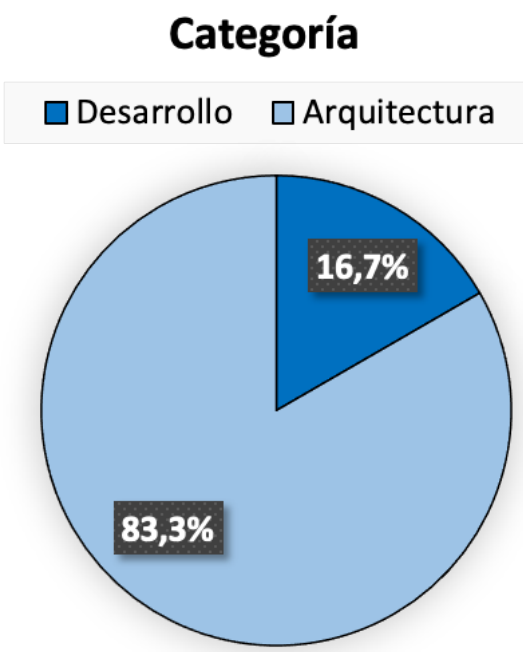
Escenario reactivo: resultados productividad

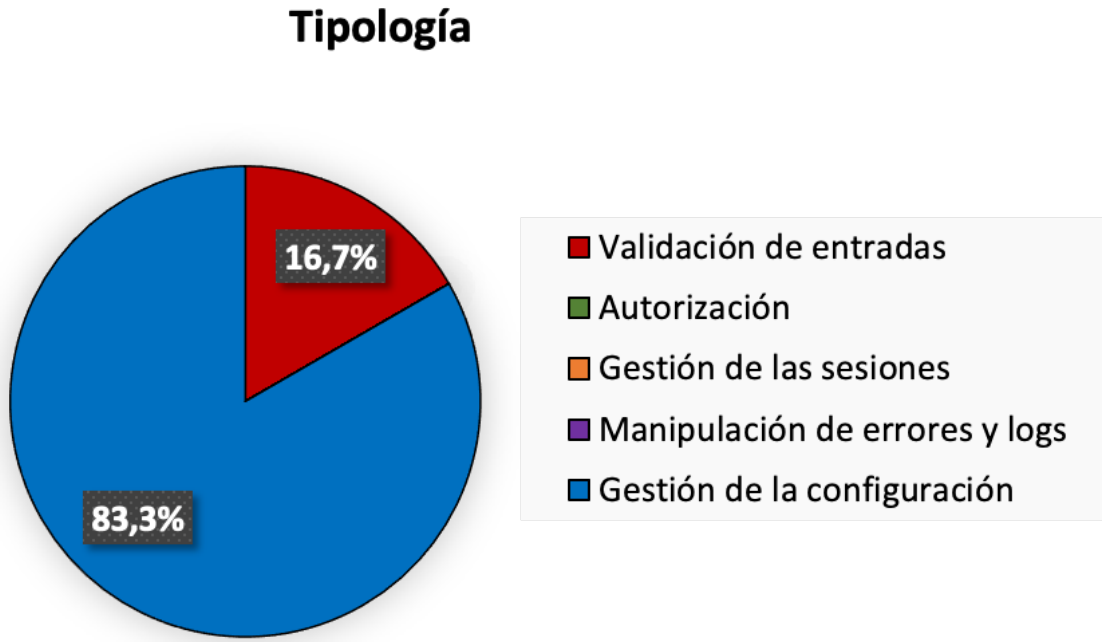


Escenario preventivo: resultados seguridad

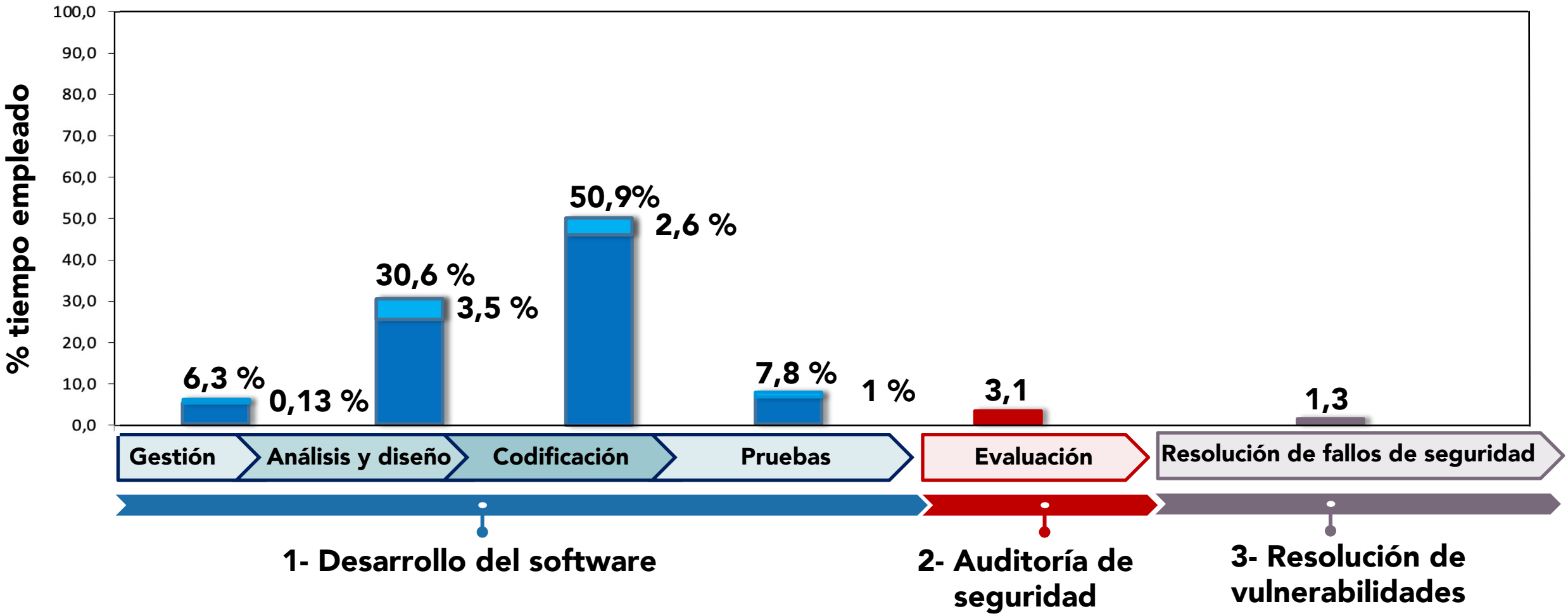
6 VULNERABILIDADES







Escenario preventivo: resultados productividad



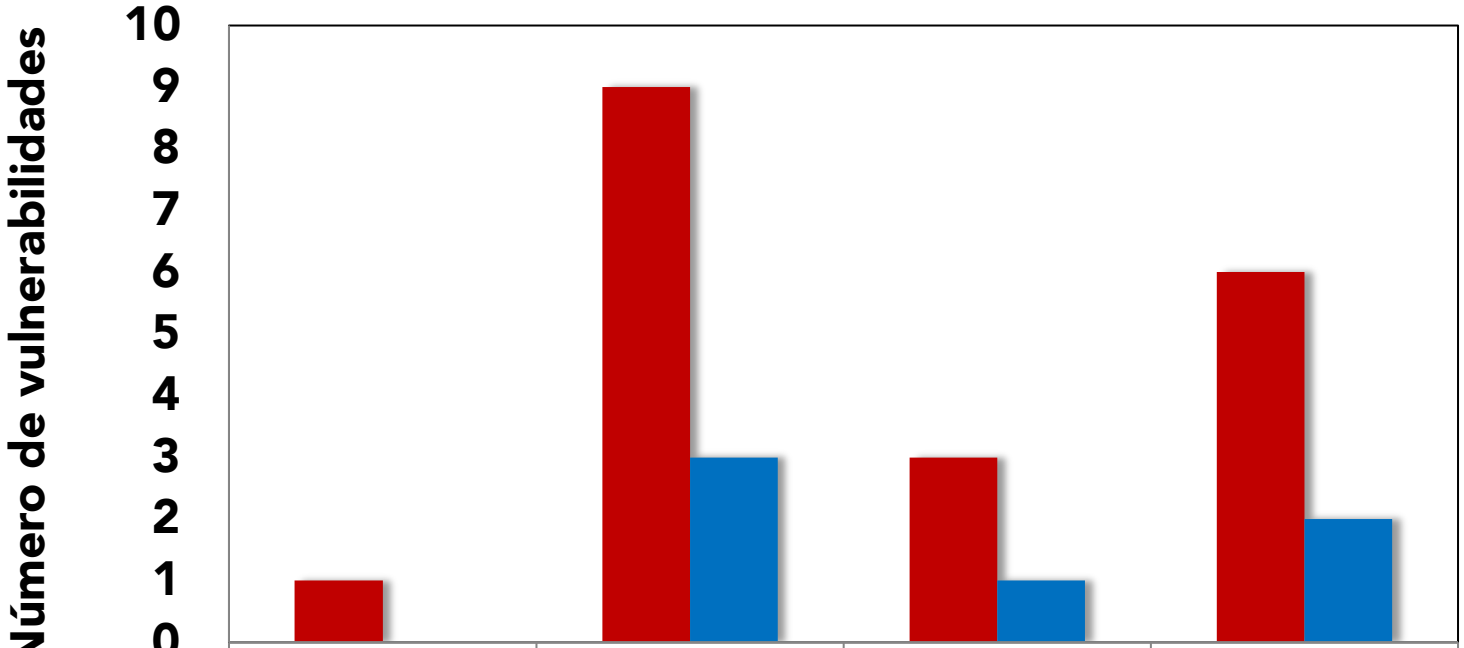
Comparativa de vulnerabilidades

TIPO	VULNERABILIDAD	CRITICIDAD	CATEGORÍA	REACTIVO	PREVENTIVO
Validación de entradas	Stored Cross Site Scripting	Crítica [9-10]	Desarrollo	X	
	Reflected Cross Site Scripting	Alta [7-8.9]	Desarrollo	X	
	Base de datos accesible	Baja [0.1-3.9]	Desarrollo	X	X
	Atributo autocomplete no inhabilitado	Baja [0.1-3.9]	Desarrollo	X	
	Cambio de peticiones POST por GET	Alta [7-8.9]	Arquitectura	X	
	Directiva POST con parámetros no validados	Alta [7-8.9]	Desarrollo	X	
Autorización	Escalada de privilegios funcional	Alta [7-8.9]	Desarrollo	X	
	Escalada de privilegios por URL	Alta [7-8.9]	Desarrollo	X	
Gestión de sesiones	Identificador de sesión desprotegido	Alta [7-8.9]	Arquitectura	X	
	Cierre de sesión no implementado correctamente	Media [4.0-6.9]	Arquitectura	X	

Comparativa de vulnerabilidades

TIPO	VULNERABILIDAD	CRITICIDAD	CATEGORÍA	REACTIVO	PREVENTIVO
Manipulación de errores y logs	Información sensible del aplicativo y uso de componentes vulnerables	Media [4.0-6.9]	Arquitectura	X	
	Información sensible en los metadatos	Baja [0.1-3.9]	Arquitectura	X	
	Información sensible en el código fuente	Media [4.0-6.9]	Arquitectura	X	
Gestión de la configuración	Página del servidor por defecto	Baja [0.1-3.9]	Arquitectura	X	
	Aplicativo en HTTP en lugar de HTTPS	Alta [7-8.9]	Arquitectura	X	
	Phising a través marcos	Alta [7-8.9]	Arquitectura	X	
	Inyección de enlaces	Alta [7-8.9]	Arquitectura	X	
	Certificados SSL débiles	Alta [7-8.9]	Arquitectura		X
	Servicios y puertos habilitados indebidamente	Alta [7-8.9]	Arquitectura		X
	Respuesta de TCP timestamp	Baja [0.1-3.9]	Arquitectura	X	X

Comparativa de resultados: seguridad



	Críticas	Altas	Medias	Bajas
Reactivo	1	9	3	6
Preventivo	0	3	1	2

VULNERABILIDADES

REACTIVO

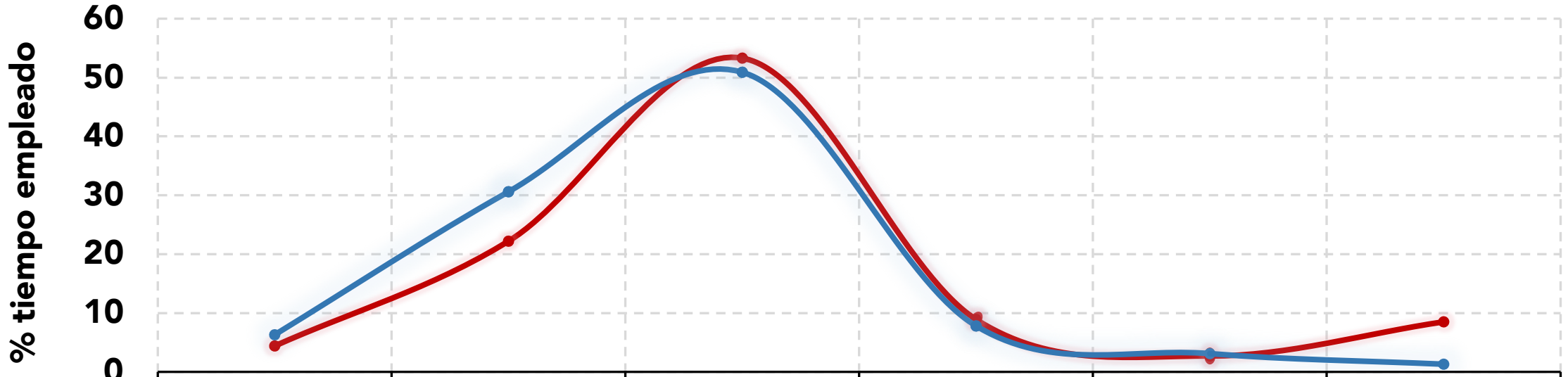
19

PREVENTIVO

6

68%

Comparativa de resultados: productividad



	Gestión	Análisis y diseño	Codificación	Pruebas	Evaluación	Corrección de vulnerabilidades
Reactivo	4,4	22,2	53,3	8,9	2,7	8,5
Preventivo	6,3	30,6	50,9	7,8	3,1	1,3

1- Desarrollo del software

2- Auditoría de seguridad

3- Resolución de vulnerabilidades

Conclusiones I

La aplicación de metodologías reactivas:

- **Genera un mayor número de vulnerabilidades.**
- **Las vulnerabilidades que se identifican tienen criticidades altas.**
- **Producen un alto impacto en la resolución de los fallos de seguridad.**

La aplicación del modelo Viewnext-UEx con enfoque preventivo:

- **Reduce drásticamente la aparición de vulnerabilidades.**
- **Reduce la criticidad de las vulnerabilidades identificadas.**
- **Minimiza el tiempo de la resolución de los fallos de seguridad.**

Conclusiones I

El modelo Viewnext-UEx que se propone:

- **Incorpora prácticas de seguridad de manera sistemática y planificada.**
- **Se ha implantado en proyectos de software reales.**
- **Conduce a una mejora en los aspectos de seguridad y optimización en los tiempos de desarrollo del ciclo de vida del software.**
- **Puede ser implantado por cualquier empresa del sector.**

Líneas futuras

Implantar el modelo en un número considerable de proyectos

SEC
DEV
OPS

Adaptar el modelo Viewnext-UEx al enfoque SecDevOps

Publicaciones muy relevantes

IEEE Access

José Carlos Sancho Núñez, Andrés Caro Lindo and Pablo García Rodríguez. A Preventive Secure Software Development Model for a software factory: a case study. IEEE Access. 8 (1):77653–77665. April 2020. DOI: 10.1109/access.2020.2989113.

(Impact Factor = 3.745 - Q1)

José Carlos Sancho Núñez, Andrés Caro Lindo, Mar Ávila Vegas and Alberto Bravo Gómez. *New approach for threat classification and security risk estimations based on security event management.* Future Generation Computer Systems. 113:488–505. December 2020. DOI: 10.1016/j.future.2020.07.015.

(Impact Factor = 6.125 - Q1)

UNIVERSIDAD DE EXTREMADURA

La Seguridad en el Desarrollo de Software implementada de forma Preventiva

José Carlos Sancho Núñez
(jcsanchon@unex.es)

Profesor Universidad de Extremadura e
Investigador Cátedra Viewnext-UEx

02/12/2021