



E.T.S. INGENIERÍA
INFORMÁTICA

UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DEL SOFTWARE

TEMA 2. ASPECTOS TRANSVERSALES



CONTENIDO

- Aspectos transversales
 - Gestión de riesgos
 - Gestión de costes
 - Gestión de la calidad
 - Gestión de las personas
 - Gobernanza y cumplimiento
 - DevOps y DevSecOps



OBJETIVOS DEL TEMA

- Comprender la importancia de la gestión de riesgos y ser capaz de llevarla a cabo.
- Entender cómo se realizan los cálculos de costes y conocer los elementos a tener en cuenta.
- Conocer el el proceso de la gestión de la calidad y las principales actividades de gestión de calidad.
- Entender el papel de las personas en el desarrollo de un proyecto y ser capaz de gestionarlas durante el mismo.
- Conocer los conceptos de gobernanza y cumplimiento en proyectos software.
- Conocer los ciclos de DevOps y DevSecOps y aplicarlos en un proyecto real.



TERMINOLOGÍA BÁSICA

- Las **actividades** en un proyecto son las divisiones del trabajo en el mismo. Se pueden dividir según su objetivo (planificación, gestión de riesgos, requisitos, calidad, etc.) o en función de su tamaño (**subproyectos, paquetes de trabajo, tareas, subtareas...**)
- Los **artefactos** o **entregables** son los productos tangibles del trabajo en el proyecto. Los **entregables** (deliverables) son resultados del proyecto que serán entregados a los clientes
- Los **recursos** son los elementos con los que contamos para desarrollar el proyecto. Normalmente, **recurso=persona**.
- Los **hitos** son puntos de control comprobables objetivamente (por ejemplo, los puntos finales de alguna actividad o el momento en el que algún artefacto está disponible)



SIMILITUDES CON LA GESTIÓN DE OTROS TIPOS DE PROYECTOS

- Las actividades de la gestión no solo son particulares en esta disciplina
- Muchas técnicas de la ingeniería de proyectos o de la investigación de operaciones son igualmente aplicables a la gestión de proyectos
- Los proyectos de ingeniería complejos tienden a sufrir los mismos problemas que los sistemas software



GESTIÓN DEL RIESGO



OBJETIVOS

- Identificar y formular riesgos de forma útil
- Analizar probabilidad, impacto e incertidumbre
- Planificar respuestas y reservas
- Monitorizar riesgos como actividad continua
- Incorporar riesgos tecnológicos, humanos, organizativos, de seguridad y de IA



DEL RIESGO A LA INCERTIDUMBRE

- Riesgo: efecto de la *incertidumbre* sobre los objetivos
- Amenazas y oportunidades
- La incertidumbre también incluye ambigüedad y complejidad
- Gestionar riesgo no significa intentar eliminar toda incertidumbre

Fuente conceptual: PMI (2021), PMBOK® Guide, 7th ed., Uncertainty Performance Domain.



UN CAMBIO IMPORTANTE EN LA GESTIÓN MODERNA

- De un “registro de riesgos” estático...
 - → a una gestión continua de incertidumbre
 - → integrada en decisiones, planificación y ejecución
 - → conectada con stakeholders, producto, arquitectura y operaciones

Fuente: ISO 31000:2018; PMI (2021).



ACTIVIDADES DEL PROCESO DE GESTIÓN DEL RIESGO

- 1. Identificación
- 2. Análisis
- 3. Planificación / tratamiento
- 4. Seguimiento y control
- El ciclo se repite durante todo el proyecto



ARTEFACTOS PRINCIPALES

- Registro de riesgos
- Valoración / mapa de riesgos
- Planes de respuesta
- Indicadores de riesgo (KRIs)
- Decisiones, responsables y fechas de revisión



ACTIVIDAD: IDENTIFICACIÓN DE RIESGOS

- Objetivo: descubrir incertidumbres que puedan afectar a los objetivos
- No basta con listar “cosas malas”
- Formular riesgo como:
causa → evento → consecuencia → acciones
- Asignar propietario desde el principio



DESCRIPCIÓN DE UN RIESGO

- Elementos mínimos:
 - CAUSA: dependencia de un proveedor
 - EVENTO: el proveedor retrasa la API
 - CONSECUENCIA: se retrasa la integración y la entrega
 - PROBABILIDAD: Alta
 - SEVERIDAD / IMPACTO: Media
- Esta formulación facilita analizar y tratar el riesgo



FUENTES DE RIESGOS

- Tecnología y arquitectura
- Personas y conocimiento
- Organización y proveedores
- Requisitos y stakeholders
- Planificación, estimaciones y dependencias
- Seguridad, privacidad y cumplimiento



NUEVAS FUENTES ESPECIALMENTE RELEVANTES

- Dependencias y software de terceros
- Cadena de suministro de software
- Servicios cloud y concentración de proveedores
- Vulnerabilidades y exposición operacional
- Datos y modelos de IA
- Uso de IA generativa durante el desarrollo
- Uso de IA generativa por adversarios



IDENTIFICACIÓN: TÉCNICAS

- Brainstorming estructurado
- Checklists y taxonomías
- Entrevistas con expertos y stakeholders
- Premortem: “el proyecto ha fracasado, ¿por qué?”
- Revisión de arquitectura y dependencias
- Lecciones aprendidas de proyectos anteriores



ANÁLISIS PREMORTEM

- Imaginar que el proyecto ha fracasado
- Identificar retrospectivamente las causas
- Convertir las causas plausibles en riesgos
- Útil para evitar el sesgo de exceso de confianza



RIESGOS DE SEGURIDAD Y SUPPLY CHAIN

- Dependencias vulnerables o abandonadas
- Compromiso de proveedores o artefactos
- Credenciales y secretos expuestos
- Ataques a CI/CD o repositorios
- Riesgos de integridad y procedencia del software

Fuente: NIST SP 800-161 Rev. 1 (actualización 2024), Cybersecurity Supply Chain Risk Management.



RIESGOS EN PROYECTOS CON IA

- Calidad y representatividad de datos
- Drift y cambios en el entorno
- Alucinaciones / comportamiento no fiable
- Privacidad, propiedad intelectual y datos sensibles
- Seguridad de modelos y prompts
- Dificultad de evaluación y monitorización

Fuente: NIST AI RMF 1.0 y NIST AI 600-1 (2024), Generative AI Profile.



ACTIVIDAD: ANÁLISIS DEL RIESGO

- Objetivo: estimar importancia para poder decidir
- Probabilidad / frecuencia
- Impacto / consecuencia
- Exposición o severidad
- Incertidumbre de la propia estimación



ANÁLISIS CUALITATIVO

- Escalas sencillas: muy bajo → muy alto
 - No más de 5 niveles
- Permite priorizar rápidamente
- Debe definirse qué significa cada nivel
- Evitar falsa precisión: “3,7” no significa necesariamente que sepamos más

Fuente: PMI (2021); ISO 31000:2018.

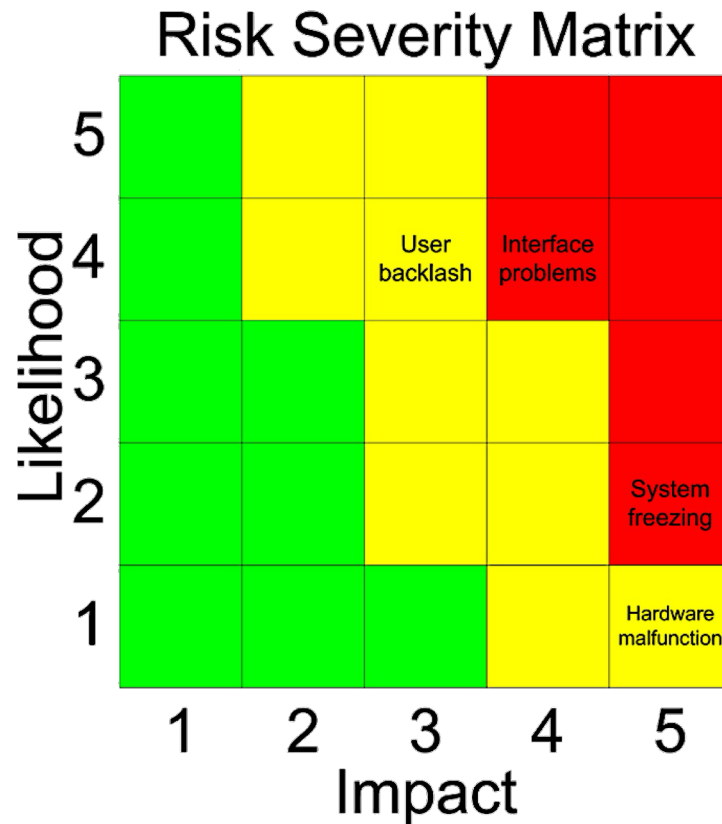


MATRIZ PROBABILIDAD-IMPACTO

- Útil como herramienta de comunicación y priorización
- No debe sustituir al juicio profesional
- Un evento raro de impacto catastrófico puede requerir más atención que muchos riesgos menores



MATRIZ PROBABILIDAD-IMPACTO



- Planear y monitorear todos los riesgos es muy costoso.
- Una matriz de severidad del riesgo puede ser útil para priorizar los riesgos.
 - La zona roja identifica los eventos más importantes
 - La zona amarilla enumera los riesgos que son moderadamente importantes
 - Los eventos de la zona verde probablemente se pueden ignorar con seguridad



ANÁLISIS CUANTITATIVO

- Cuando existe información suficiente:
 - Expected Monetary Value (EMV)
 - simulación Monte Carlo
 - distribuciones de esfuerzo / duración
 - análisis de sensibilidad
 - CORAS
- Útil para reservas y decisiones de inversión



RIESGO INDIVIDUAL VS. RIESGO GLOBAL

- Riesgos individuales: eventos concretos
- Riesgo global: efecto de la incertidumbre sobre el proyecto como sistema
- Las interacciones entre riesgos pueden ser importantes
- Ej.: retraso del proveedor + pérdida de personal + cambio de requisitos

Fuente: PMI (2021), enfoque de incertidumbre y análisis cuantitativo.



ACTIVIDAD: PLANIFICACIÓN / TRATAMIENTO

- Para cada riesgo significativo:
- Evitar
- Mitigar / reducir
- Transferir / compartir
- Aceptar
- Escalar cuando corresponda

Fuente: PMI (2021), Uncertainty Performance Domain.



EVITAR, MITIGAR, TRANSFERIR, ACEPTAR

- **EVITAR:**
 - cambiar el plan para eliminar la amenaza
- **MITIGAR:**
 - reducir probabilidad o impacto
- **TRANSFERIR:**
 - trasladar parte de la exposición
- **ACEPTAR:**
 - asumirla conscientemente y preparar respuesta



OPORTUNIDADES

- El riesgo no es solo amenaza
 - Explotar → asegurar que ocurra
 - Mejorar → aumentar probabilidad o impacto positivo
 - Compartir → colaborar para capturar beneficio
 - Aceptar → aprovechar si ocurre



CONTINGENCIA $\neq$ MITIGACIÓN

- Mitigación: actuar antes del evento
- Contingencia: actuar si el evento ocurre
- Ejemplo:
 - Mitigar $\rightarrow$ prototipo temprano de una integración
 - Contingencia $\rightarrow$ alternativa técnica preparada



RESERVAS Y TOLERANCIAS

- No todos los riesgos se pueden eliminar
- Reserva de contingencia: incertidumbre conocida
- Reserva de gestión: incertidumbre no asignada / cambios fuera de la línea base
- Definir tolerancias evita reaccionar tarde



PROPIETARIO DEL RIESGO

- Cada riesgo relevante debe tener un responsable
- Responsable $\neq$ persona culpable
- Su función es vigilar señales, impulsar respuestas y actualizar el estado
- Sin propietario, el riesgo tiende a quedar “huérfano” y pasa a ser impredecible



ACTIVIDAD: SEGUIMIENTO DEL RIESGO

- Revisar probabilidad e impacto
- Comprobar si aparecen indicios (señales tempranas)
- Actualizar respuestas y reservas
- Cerrar riesgos que ya no sean relevantes
- Identificar nuevos riesgos



INDICADORES CLAVE DE RIESGO - KRIS

- Un KRI es una señal que anticipa un aumento de exposición
- Ejemplos:
 - rotación del equipo
 - defectos abiertos críticos
 - dependencias sin mantenimiento
 - retraso de hitos
 - incidentes de seguridad
 - cambios frecuentes de requisitos



RISK BURNDOWN / TENDENCIA

- Más importante que tener una fotografía estática es poder observar la evolución
 - ¿Está disminuyendo la exposición total?
 - ¿Qué riesgos están creciendo?
 - ¿Cuántos riesgos críticos permanecen abiertos?
- Las tendencias ayudan a decidir antes de que el problema se materialice



GESTIÓN DE RIESGOS EN AGILE

- El riesgo no desaparece con Scrum
- Se integra en backlog, refinamiento, planificación y retrospectivas
- Los spikes/prototipos pueden reducir incertidumbre
- Priorizar riesgos técnicos puede adelantar aprendizaje



GESTIÓN DE RIESGOS EN DEVOPS Y OPERACIÓN

- La exposición continúa después del desarrollo
- Observabilidad → detección temprana
- Automatización → reducción de errores humanos
- Despliegues pequeños → menor impacto potencial
- Rollback / recuperación → limitar consecuencias



RIESGO DE TERCEROS Y SUPPLY CHAIN

- Inventariar componentes y proveedores
- Evaluar criticidad y dependencia
- Monitorizar vulnerabilidades y mantenimiento
- Verificar procedencia e integridad cuando sea necesario
- Preparar alternativas para dependencias críticas



RIESGO Y SOFTWARE SEGURO

- Integrar seguridad desde el diseño
- Threat modeling y análisis de arquitectura
- Requisitos de seguridad trazables
- Verificación continua en CI/CD
- Gestionar vulnerabilidades como riesgo, no solo como lista de fallos

Fuente: NIST CSF 2.0 (2024); NIST SP 800-161 Rev. 1.



RIESGO Y SISTEMAS DE IA

- Gestionar el riesgo durante todo el ciclo de vida
- Identificar usos, actores, impactos y límites
- Evaluar seguridad, robustez, privacidad y sesgos
- Monitorizar comportamiento después del despliegue
- Reevaluar cuando cambian datos, modelo o contexto

Fuente: NIST CSF 2.0 (2024); NIST SSDF Community Profile for Generative AI (2024).



EJEMPLO: RIESGO DE IA GENERATIVA

- Riesgo: el asistente genera código inseguro
- Causa: confianza excesiva en la salida
- Consecuencia: vulnerabilidad introducida en producción
- Mitigación: revisión humana + análisis automatizado + tests de seguridad
- KRI: defectos/vulnerabilidades detectados en código generado

Fuente: NIST AI RMF 1.0; NIST AI 600-1 (2024).



REGISTRO DE RIESGOS: ESTRUCTURA RECOMENDADA

- ID · descripción · causa · consecuencia
- Probabilidad · impacto · exposición
- Propietario · respuesta · acciones
- KRI / trigger · fecha de revisión
- Estado: abierto / en tratamiento / materializado / cerrado



DE “LISTA DE RIESGOS” A DECISIONES

- Un buen registro no es un inventario burocrático
- Cada riesgo importante debe conducir a una decisión:
 - ¿qué hacemos?
 - ¿quién lo hace?
 - ¿cuándo?
 - ¿cómo sabremos que la exposición ha cambiado?



INTEGRACIÓN CON LA GESTIÓN DEL PROYECTO





ERRORES FRECUENTES

- Identificar riesgos una sola vez
- Registrar riesgos sin propietario
- Confundir riesgo con problema ya ocurrido
- Usar matrices sin criterios definidos
- No actualizar probabilidades/impactos
- Ignorar riesgos positivos
- Tratar riesgos técnicos como asunto exclusivo del equipo técnico



RESUMEN

- Identificar → analizar → tratar → monitorizar
- Formular riesgos como causa–evento–consecuencia
- Priorizar exposición, no cantidad de riesgos
- Usar señales tempranas y tendencias
- Integrar seguridad, supply chain e IA
- El objetivo es mejorar decisiones bajo incertidumbre



REFERENCIAS RIESGOS

- PMI (2021). A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 7th ed. — Uncertainty Performance Domain
- ISO 31000:2018. Risk management — Guidelines
- IEC 31010:2019. Risk management — Risk assessment techniques
- Hillson, D. (2009). Managing Risk in Projects. Gower
- NIST (2024). Cybersecurity Framework (CSF) 2.0
- NIST SP 800-161 Rev. 1, actualización 2024. Cybersecurity Supply Chain Risk Management
- NIST (2024). AI RMF: Generative AI Profile (NIST AI 600-1)
- NIST (2024). Secure Software Development Practices for Generative AI and Dual-Use Foundation Models

PMI (2021); ISO 31000:2018; IEC 31010:2019; Hillson (2009).

NIST CSF 2.0 (2024); NIST SP 800-161 Rev. 1 (2024 update); NIST AI 600-1 (2024); NIST SSDF Generative AI Community Profile (2024).



GESTIÓN DE COSTES



¿QUÉ SIGNIFICA GESTIONAR EL COSTE?

- **Objetivo**
 - **Asegurar que el proyecto utiliza los recursos económicos de forma sostenible para conseguir el valor esperado.**
- **Actividades principales**
 - **Estimar costes**
 - **Presupuestar el proyecto**
 - **Asignar y controlar recursos**
 - **Medir el coste real y el progreso**
 - **Prever el coste final**
 - **Tomar decisiones ante desviaciones**

Project Management Institute (2021). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 7th ed.



¿CUÁLES SON LOS COSTES?

- Personas → principal coste en muchos proyectos software
- Infraestructura y herramientas
 - Cloud y servicios externos
- Pruebas, integración y despliegue
- Seguridad y cumplimiento
- Mantenimiento y operación
- Cambios y trabajo
- El coste de desarrollar una funcionalidad no es necesariamente su coste total de propiedad (TCO).

Project Management Institute (2021). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 7th ed.



ESTIMACIÓN DEL COSTE EN DESARROLLO TRADICIONAL

En un proyecto **predictivo**, normalmente intentamos estimar el coste global antes de comenzar la ejecución.

Trabajo → esfuerzo → recursos → coste

Presupuesto ejemplo:

Recurso	Esfuerzo	Coste/h	Coste
Analista	120 h	40 €	4.800 €
Desarrollador	600 h	35 €	21.000 €
Tester	180 h	30 €	5.400 €
Infraestructura	—	—	3.000 €
Total			34.200 €

Pero... estimación ≠ presupuesto ≠ coste real: La incertidumbre debe reflejarse mediante **reservas y contingencias**.

Hughes, B., Cotterell, M., & Mall, R. (2017). *Software Project Management*, 6th ed. McGraw-Hill.



CONTROL DEL COSTE

- El presupuesto sólo es útil si podemos comprobar cómo evoluciona el proyecto respecto a lo previsto. Preguntas:
 - ¿Cuánto trabajo hemos realizado?
 - ¿Cuánto nos ha costado?
 - ¿Cuánto deberíamos haber gastado?
 - ¿Cuánto esperamos gastar finalmente?



EARNED VALUE MANAGEMENT (EVM)

- Una herramienta clásica es EMV:
 - PV — Planned Value
 - EV — Earned Value
 - AC — Actual Cost
- Calcula dos índices que permiten saber si el proyecto está funcionando según lo previsto en términos de **coste** y **plazo**.
- **CPI** (Cost Performance Index) que mide la eficiencia del coste como EV/AC
- **SPI** (Schedule Performance Index) que mide la eficiencia respecto al calendario como EV/PV
- En software, estas métricas deben interpretarse con cuidado: medir “porcentaje completado” de funcionalidades puede generar una falsa sensación de precisión.

Fleming, Q. W., & Koppelman, J. M. (2016). *Earned Value Project Management*, 4th ed. PMI.



ESTIMACIÓN DEL COSTE EN DESARROLLO AGILE

- En Agile, normalmente **no intentamos fijar simultáneamente alcance, plazo y coste con precisión desde el principio. Se fija relativamente**
Equipo + duración → capacidad/coste
 - y se adaptan el alcance y las prioridades
- **Ejemplo**
 - Si un equipo de **6 personas** tiene un coste mensual de **30.000 €**
 - En un sprint de **2 semanas** el coste aproximado del sprint es de **15.000 €**
- La clave es el cambio de pensamiento.
 - En vez de “¿Cuánto costará exactamente todo el producto?” pasamos a “¿**Qué valor podemos obtener con los recursos disponibles?**”
- La prioridad del backlog determina **qué funcionalidades se abordan primero.**

Cohn, M. (2005). *Agile Estimating and Planning*. Prentice Hall.



COSTE Y DECISIONES: TRADICIONAL VS. AGILE

- En ambos casos el coste no considera únicamente el desarrollo:
Desarrollo + calidad + seguridad + infraestructura + operación + mantenimiento + deuda técnica
- La gestión del coste consiste en tomar decisiones económicas informadas, no simplemente en gastar menos.

	Tradicional / predictivo	Agile / adaptativo
Presupuesto	Se estima inicialmente	Se relaciona con equipo y horizonte
Alcance	Intenta fijarse	Se prioriza y adapta
Planificación	Detallada inicialmente	Rolling wave / iterativa
Estimación	Horas, recursos, actividades	Story points, capacidad, velocidad
Control	Coste planificado vs. real	Coste por iteración + valor entregado
Cambios	Analizar impacto en presupuesto/plazo	Re-priorizar backlog
Pregunta clave	“¿Nos ajustamos al presupuesto?”	“¿Estamos obteniendo suficiente valor?”

Project Management Institute (2021). *PMBOK® Guide*, 7th ed.

Cohn, M. (2005). *Agile Estimating and Planning*.

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate*. IT Revolution.



GESTIÓN DE LA CALIDAD



GESTIÓN DE CALIDAD

- Se centra en asegurar que el nivel de calidad requerido se alcanza en un producto software.
- Implica la definición de normas (estándares internos) de calidad adecuadas y los procedimientos para garantizar que estas se siguen.
- Debe tratar de crear una "cultura de calidad", donde la calidad sea vista como una responsabilidad de todos.



POR QUÉ ES IMPORTANTE LA CALIDAD DEL CÓDIGO

- La calidad del código influye en:
 - La legibilidad de los programas
 - Reducir la probabilidad de cometer errores
 - Facilitar la búsqueda de errores
 - El mantenimiento y evolución del software
 - Trabajo colaborativo
 - Reusabilidad
 - Seguridad



DEFINICIÓN DE CALIDAD

- La Calidad software, simplificando, significa que un producto debe cumplir con sus especificaciones.
- Esto no es fácil de conseguir o comprobar para los sistemas software
 - Hay una tensión entre los requisitos de calidad de los clientes (eficiencia, fiabilidad, privacidad, etc.) y de los desarrolladores (mantenimiento, reutilización, etc.)
 - Algunos requisitos de calidad son difíciles de especificar de manera inequívoca;
 - Las especificaciones generalmente son incompletas y, a menudo, tienen incoherencias.



ACTIVIDADES

- Aseguramiento/Garantía de la calidad
 - Establecer procedimientos de organización y estándares de calidad.
- Planificación de la calidad
 - Seleccionar los procedimientos y estándares para un proyecto en particular y modificarlos según las necesidades
- Control de la calidad
 - Garantizar que los procedimientos y estándares son seguidos por el equipo de desarrollo de software.
- La gestión de la calidad debe estar separada de la gestión del proyecto para garantizar su independencia.



DEFINICIÓN DE CALIDAD

- Pero en la práctica, ¿qué es la calidad de un código?
 - Determinar si un código tiene una calidad alta o baja puede ser subjetivo
- Se supone que un código de calidad
 - hace lo que debe (tanto en funciones de negocio como de seguridad)
 - sigue un estilo consistente
 - es fácil de entender
 - está bien documentado
 - se puede probar
 - **no tiene vulnerabilidades**
 - **no puede usarse de formas no previstas**



DEFINICIÓN DE CALIDAD

Bjarne Stroustrup, inventor of C++ and author of *The C++ Programming Language*

I like my code to be elegant and efficient. The logic should be straightforward to make it hard for bugs to hide, the dependencies minimal to ease maintenance, error handling complete according to an articulated strategy, and performance close to optimal so as not to tempt people to make the code messy with unprincipled optimizations. Clean code does one thing well.

“Big” Dave Thomas, founder of OTI, godfather of the Eclipse strategy

Clean code can be read, and enhanced by a developer other than its original author. It has unit and acceptance tests. It has meaningful names. It provides one way rather than many ways for doing one thing. It has minimal dependencies, which are explicitly defined, and provides a clear and minimal API. Code should be literate since depending on the language, not all necessary information can be expressed clearly in code alone.

Clean Code: A Handbook of Agile Software Craftsmanship



DEFINICIÓN DE CALIDAD

Michael Feathers, author of *Working Effectively with Legacy Code*

I could list all of the qualities that I notice in clean code, but there is one overarching quality that leads to all of them. Clean code always looks like it was written by someone who cares. There is nothing obvious that you can do to make it better. All of those things were thought about by the code's author, and if you try to imagine improvements, you're led back to where you are, sitting in appreciation of the code someone left for you—code left by someone who cares deeply about the craft.

Clean Code: A Handbook of Agile Software Craftsmanship



EVOLUCIÓN HISTÓRICA

- La calidad comenzó siendo sinónimo de la **falta de defectos**.
- Hoy se persigue el concepto de **Gestión Total de la Calidad** (abreviada TQM, del inglés *Total Quality Management*)
 - El concepto de calidad total está relacionado con el de la “percepción de la calidad”
 - A pesar de que surgió en los 50, no fue hasta mediados de los 80 cuando empezó a popularizarse
 - Los procesos de SQA (Software Quality Assurance) se inspiran en TQM en su concepto holístico de la calidad.



CALIDAD Y SEGURIDAD: DOS CARAS DE LA MISMA MONEDA

- **Conceptos clave**

- **Calidad del software:** grado en el que un sistema cumple con los requisitos funcionales y no funcionales (ISO/IEC 25010).
- **Seguridad del software:** capacidad del producto para proteger información y datos de acceso o modificación no autorizada.

- **Relación fundamental**

- La **seguridad es una subcaracterística de la calidad** dentro del modelo ISO/IEC 25010 → pertenece al atributo “*fiabilidad*” y “*protección de la información*”.
- Un software **no puede considerarse de alta calidad si no es seguro**, aunque cumpla su funcionalidad.

Fuente: ISO/IEC 25010:2011 — *Systems and Software Quality Models*.

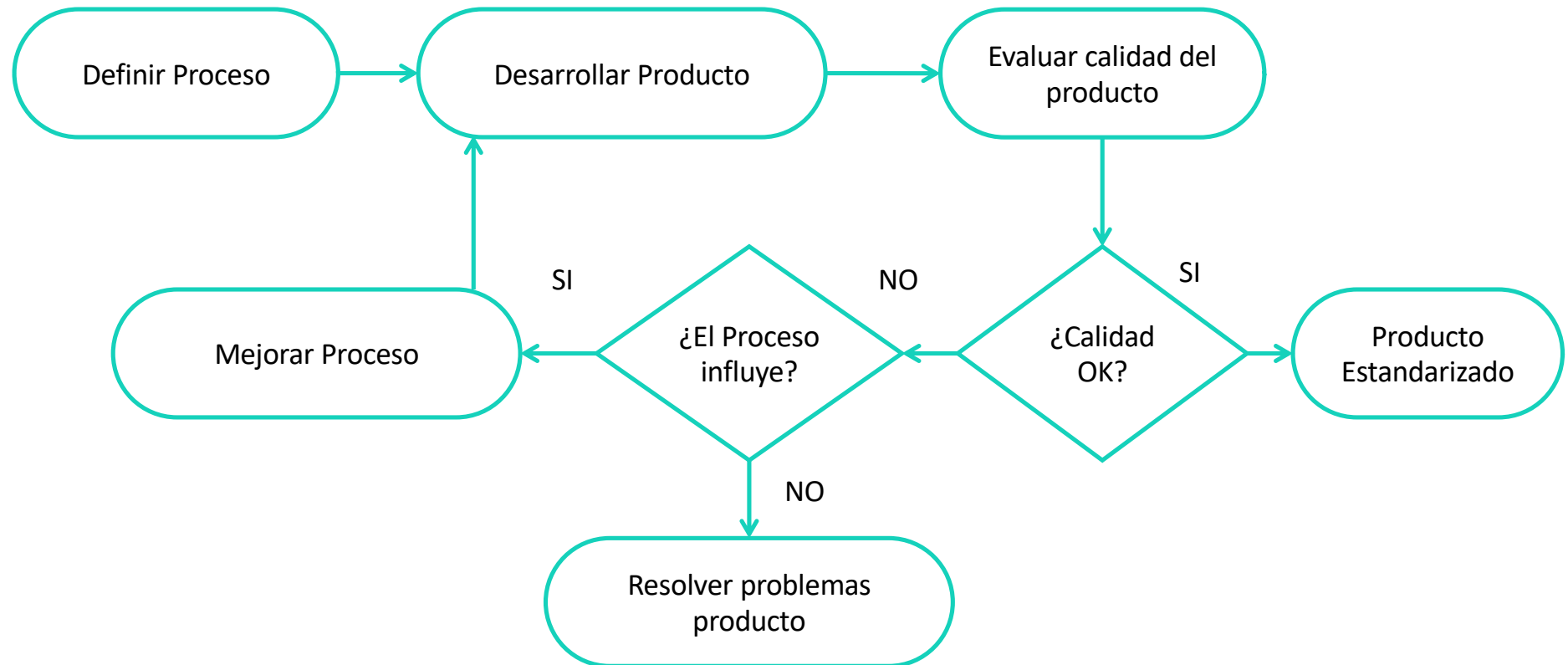


RELACIÓN ENTRE CALIDAD DE PROCESO Y DE PRODUCTO

- **Hipótesis:** La calidad de un producto desarrollado está influida por la calidad del proceso de su producción.
 - Esto es importante en el desarrollo de software, ya que algunos atributos de la calidad de los productos son difíciles de evaluar.
- Sin embargo, existe una relación muy compleja y difícil de entender entre el proceso de desarrollo del software y la calidad del producto. Además, esta relación es aún más compleja en los aspectos de seguridad.



CALIDAD DE PROCESO / PRODUCTO





GESTIÓN DE LAS PERSONAS



PERSONAL DEL PROYECTO

- Puede ser imposible reclutar a las personas ideales para trabajar en el proyecto
 - El presupuesto del proyecto no permite pagar altos salarios de personal experimentado
 - El personal con la experiencia necesaria puede que no esté disponible
 - La organización puede preferir formar a sus emplead@s en las capacidades necesarias del desarrollo de proyectos de software
- El personal de gestión tiene que trabajar con estas limitaciones, especialmente cuando hay escasez de personal capacitado



LAS PERSONAS SON LA CLAVE DEL PROYECTO

- El software lo construyen personas.
 - Los generadores de código (como la IA) siguen siendo controlados por las personas, que deciden si aceptan o no el código generado.
- La planificación puede ser excelente y el proyecto fracasar por problemas de personas.
 - Y esto sucede muchas veces ...



¿QUÉ DEBE GESTIONAR UN RESPONSABLE DE PROYECTO?

- **Capacidad:** ¿cuánto trabajo puede asumir cada persona?
- **Competencias:** ¿qué conocimientos necesita cada tarea?
- **Coordinación:** ¿quién necesita colaborar con quién?
- **Motivación:** ¿qué favorece compromiso, autonomía y responsabilidad?
- **Comunicación:** ¿qué información necesita cada persona y cuándo?
- **Desarrollo:** ¿cómo evoluciona el equipo durante el proyecto?
- La persona responsable de proyecto debe entender que está al servicio del equipo. No al contrario.



CONOCER AL EQUIPO ES EL PRIMER PASO

ORIENTACIÓN A TAREAS

Se centra en **conseguir resultados**.

- planificación
- precisión
- resolución de problemas
- cumplimiento

ORIENTACIÓN A INTERACCIÓN

Se centra en las **relaciones**.

- comunicación
- coordinación
- cohesión
- mediación

ORIENTACIÓN A SÍ MISMO

Se centra en **necesidades y reconocimiento** propios.

- autonomía
- influencia
- reconocimiento
- objetivos personales

- No son tipos excluyentes: una misma persona puede mostrar diferentes grados de las tres orientaciones.



¿PARA QUÉ SIRVE CONOCER LOS PERFILES?

- Evitar **formar equipos homogéneos**: la diversidad de orientaciones siempre es útil.
- **Asignar responsabilidades** teniendo en cuenta necesidades de la tarea y del equipo.
- **Detectar desequilibrios**: mucha orientación a tareas puede descuidar la coordinación; mucha orientación a interacción puede ralentizar decisiones.
- **Gestionar expectativas**: motivadores diferentes requieren formas diferentes de reconocimiento.
- **Observar conductas**, no “encasillar” personas.



GESTIONAR LA CARGA DE TRABAJO DE CADA PERSONA

- Capacidad disponible $\neq$ horas laborales totales.
 - Considerar reuniones, coordinación, soporte, formación, investigación, incidencias y otras obligaciones.
- Incluso con lo anterior, hay que evitar asignar el 100% de la capacidad: deja margen para imprevistos.
- Repartir trabajo según capacidad, competencias y dependencias.
- Revisar la carga durante el proyecto: la asignación inicial deja de ser válida.



FORMAS DE GESTIONAR Y EQUILIBRAR LA CARGA

- No existe una única técnica:
 - La elección depende del ciclo de vida, del equipo y del tipo de trabajo.

Enfoque	Cómo funciona	Riesgo / precauciones
Asignación por horas	Distribuir esfuerzo estimado según capacidad.	Las horas estimadas pueden ser inciertas.
Capacidad por iteración	Seleccionar trabajo hasta la capacidad del equipo.	No confundir velocidad con capacidad futura.
WIP limitado	Limitar tareas simultáneas por persona/equipo.	Puede requerir priorizar o esperar.
Carga visible	Usar tablero, calendario o dashboard para detectar saturación.	La herramienta no sustituye la conversación.



PROBLEMAS HABITUALES RELACIONADOS CON EL PERSONAL

- Los problemas de personal son también riesgos.
 - Falta de personas o perfiles críticos.
 - Rotación, bajas o pérdida de conocimiento.
 - Sobreasignación y multitarea excesiva.
 - Dependencia de una única persona («bus factor» bajo).
 - Conflictos, mala comunicación o falta de coordinación.
 - Falta de competencias o curva de aprendizaje mayor de la prevista.
 - Desmotivación, agotamiento o sensación de falta de autonomía.
 - Distribución desigual del trabajo o responsabilidades poco claras.



EL PAPEL DEL RESPONSABLE DEL PROYECTO

- **Prever:** identificar perfiles críticos, dependencias y riesgos de personal.
- **Visualizar:** hacer visible carga, bloqueos, WIP y responsabilidades.
- **Priorizar:** reducir trabajo simultáneo antes que saturar al equipo.
- **Redistribuir:** mover tareas, incorporar apoyo o cambiar prioridades.
- **Compartir conocimiento:** revisión de código, pair programming, documentación y rotación.
- **Comunicar pronto:** una sobrecarga detectada tarde suele convertirse en retraso.
- **Escalar** cuando sea necesario: recursos, calendario, alcance o expectativas.



MOTIVACIÓN, LIDERAZGO Y COMUNICACIÓN

- **Objetivos claros:** cada persona debe saber qué se espera y por qué.
- **Autonomía:** asignar responsabilidad sin microgestión (dejar libertad de realización de la responsabilidad).
- **Feedback frecuente:** reconocer avances y corregir problemas pronto.
- **Seguridad psicológica:** poder comunicar errores, dudas y riesgos sin temor.
- **Reconocimiento:** valorar contribuciones técnicas y de coordinación.
- **Conflictos:** abordarlos sobre hechos, necesidades y objetivos del proyecto.



RESUMEN

- Gestionar personas es gestionar capacidad y relaciones
- El rendimiento de un equipo no es la suma simple de rendimientos individuales.
- La composición del equipo importa tanto como el número de personas.
- La carga debe gestionarse continuamente.
- Los problemas de personal deben tratarse como riesgos del proyecto.
- La IA y las herramientas de gestión pueden ayudar, pero no sustituyen experiencia, liderazgo, comunicación ni decisiones humanas.



GOBERNANZA Y CUMPLIMIENTO



GOBERNANZA Y CUMPLIMIENTO

- **Idea central:** gobernanza $\neq$ gestión.
 - **Gobernanza:** establece quién tiene autoridad, qué decisiones deben tomarse y bajo qué reglas.
 - **Gestión:** ejecuta el proyecto dentro de ese marco.
- Define:
 - responsabilidades
 - autoridad y escalado
 - mecanismos de decisión
 - controles
 - transparencia y rendición de cuentas
- **La gobernanza define las reglas del juego; la gestión juega el partido.**



NECESIDAD DE GOBERNANZA

- Un proyecto implica decisiones que pueden afectar a:
 - negocio y usuarios
 - presupuesto y recursos
 - seguridad y privacidad
 - datos
 - propiedad intelectual
 - proveedores
 - calidad y operación
 - cumplimiento normativo
- **Sin gobernanza no hay responsabilidad:**
 - decisiones dispersas
 - responsabilidades difusas
 - difícil saber quién decidió y sobre todo por qué.



ELEMENTOS BÁSICOS DE LA GOBERNANZA

Podemos visualizarla como cinco piezas:

AUTORIDAD → RESPONSABILIDADES → DECISIONES → CONTROLES → INFORMACIÓN

- ¿Quién puede aprobar?
- ¿Quién ejecuta?
- ¿Qué decisiones requieren escalado?
- ¿Qué debe verificarse?
- ¿Qué información necesita cada responsable?

Ejemplos:

Decisión	Responsable
Cambio de alcance	Sponsor / comité
Decisión arquitectónica	Equipo técnico
Aceptación del producto	Cliente/Product Owner
Riesgo crítico	PM + responsable correspondiente
Tratamiento de datos	Responsable designado



GOBERNANZA NO SIGNIFICA BUROCRACIA

- **Mala gobernanza:** “Para cambiar una línea de código necesitamos tres reuniones y cuatro firmas.”
- **Buena gobernanza:** “Las decisiones importantes tienen responsable, criterios, evidencia y trazabilidad.”
- Principios:
 - proporcionalidad
 - claridad
 - transparencia
 - trazabilidad
 - responsabilidad
 - adaptación al riesgo
- Idea central: **Más controles ≠ mejor gobernanza.**



¿QUÉ ES CUMPLIMIENTO (COMPLIANCE)?

- **Compliance = cumplimiento de obligaciones aplicables al proyecto y al producto.**
- Pueden proceder de:
- **LEGALES**
 - protección de datos
 - propiedad intelectual
 - normativa sectorial
 - IA
- **DE CONTRATO**
 - requisitos del cliente
 - SLA
 - licencias
 - condiciones de proveedores
- **DE ORGANIZACIÓN**
 - políticas internas
 - procedimientos
 - estándares
- **TÉCNICAS**
 - estándares
 - requisitos de seguridad
 - controles de calidad
- **Cumplir no es simplemente “pasar una auditoría”.**



ABORDANDO EL CUMPLIMIENTO DESDE EL PRINCIPIO

- Requisito → Diseño → Implementación → Evidencia → Verificación
- Por ejemplo: “**Los datos personales deben estar protegidos.**” requiere plantear desde el principio
 - ¿Qué exige exactamente?
 - ¿Qué decisión arquitectónica lo garantiza?
 - ¿Cómo se implementa?
 - ¿Qué evidencia demuestra el cumplimiento?
 - ¿Cómo se verifica?
- Esto conecta directamente con la gestión del proyecto en **requisitos, trazabilidad, arquitectura, pruebas y calidad.**



CONSECUENCIAS DEL INCUMPLIMIENTO

- El incumplimiento puede provocar:
 - Retrasos
 - Repetición de trabajo
 - Costes adicionales
 - Bloqueo de una entrega
 - Incidentes de seguridad
 - Problemas contractuales
 - Daño reputacional
- Por eso **Compliance** → **riesgo del proyecto**
- No debería considerarse únicamente al final.



GOBERNANZA EN PROYECTOS ÁGILES Y DEVOPS

- La gobernanza **no desaparece** al trabajar con Agile. Cambia su forma:
 - **Antes:** Plan → aprobación → desarrollo → auditoría
 - **Ahora:** Planificación continua → desarrollo → controles automatizados → feedback → decisión
- Ejemplos:
 - *Definition of Done* con requisitos de seguridad
 - revisiones de código
 - controles de calidad automáticos
 - análisis de dependencias
 - CI/CD con gates
 - evidencias generadas automáticamente
 - auditoría y trazabilidad
- **La automatización DevSecOps permite integrar chequeos y pruebas de compliance en el flujo de desarrollo.**



BIBLIOGRAFÍA GOBERNANZA Y CUMPLIMIENTO

- **PMI (2021).** *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 7th ed.
- **ISO/IEC 38500 (2024).** *Governance of IT for the organization.*
- **ISO 31000 (2018).** *Risk Management — Guidelines.*
- **ISO/IEC 27001 (2022).** *Information security, cybersecurity and privacy protection — Information security management systems.*
- **ISO/IEC 27701 (2019).** *Security techniques — Privacy information management.*
- **Sommerville, I. (2016).** *Software Engineering*, 10th ed.
- **Pressman, R. S. & Maxim, B. R. (2020).** *Software Engineering: A Practitioner's Approach*, 9th ed.
- Otras normas relevantes son **RGPD, NIS2 y AI Act.**



DEVOPS Y DEVSECOPS



¿QUÉ ES DEVOPS?

DevOps = desarrollo + operaciones

- Integrar **personas, procesos y herramientas**
- Automatizar el ciclo **build → test → release → deploy → operate**
- Reducir el tiempo entre una modificación y su **entrega fiable**
- Feedback continuo entre desarrollo y operación
- Responsabilidad compartida sobre el software en producción

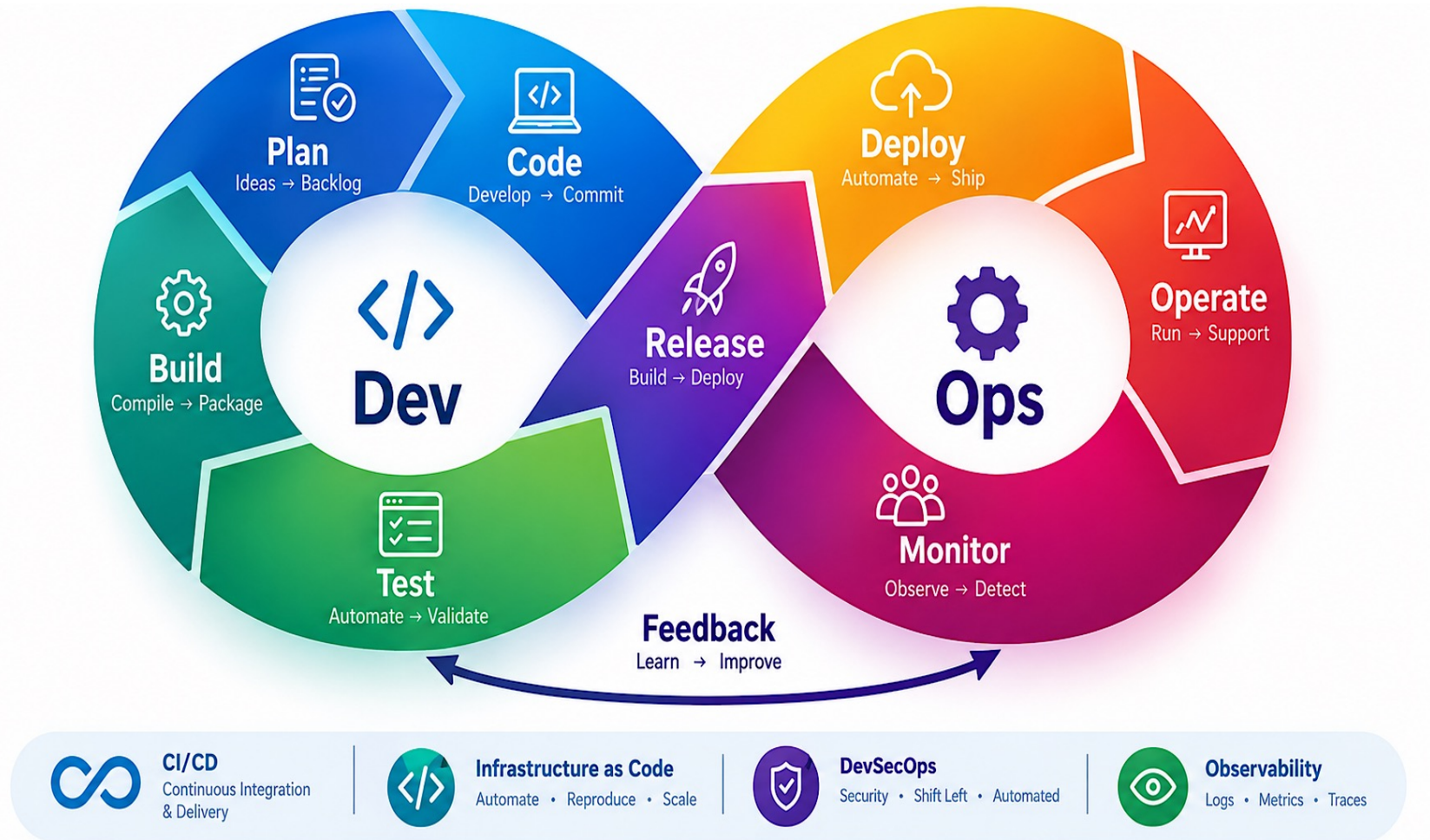
Idea clave:

DevOps no es una herramienta ni un puesto de trabajo: es una **forma de organizar el desarrollo y la operación como un flujo continuo.**

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook*. IT Revolution.



EL CICLO DevOps





EL CICLO DevOps

Automatización + integración continua + entrega continua

- **CI (Continuous Integration)**
Integrar cambios frecuentemente y validarlos automáticamente.
- **CD (Continuous Delivery/Deployment)**
Mantener el software preparado para ser entregado o desplegarlo automáticamente.
- **Infrastructure as Code**
Gestionar infraestructura mediante código y automatización.
- **Observabilidad**
Obtener información del sistema mediante logs, métricas y trazas.

Humble, J., & Farley, D. (2010). *Continuous Delivery*. Addison-Wesley.



DevOps Y GESTIÓN DEL PROYECTO

DevOps cambia la forma de gestionar el proyecto

Gestión tradicional	DevOps
Entrega al final	Entrega frecuente
Desarrollo → Operaciones	Desarrollo + Operaciones
Pruebas tardías	Pruebas automatizadas continuas
Cambios controlados por fases	Feedback continuo
Estado del proyecto	Estado del producto en producción
Métricas de actividad	Métricas de flujo y entrega

Métricas DORA

- **Deployment Frequency** → frecuencia de despliegue
- **Lead Time for Changes** → tiempo hasta entregar un cambio
- **Change Failure Rate** → cambios que provocan fallos
- **Time to Restore** → tiempo para recuperar el servicio
- **Objetivo:** entregar cambios rápidamente *sin sacrificar estabilidad ni calidad*.
- **Medir para mejorar el flujo, no para medir a las personas.**

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate*. IT Revolution.

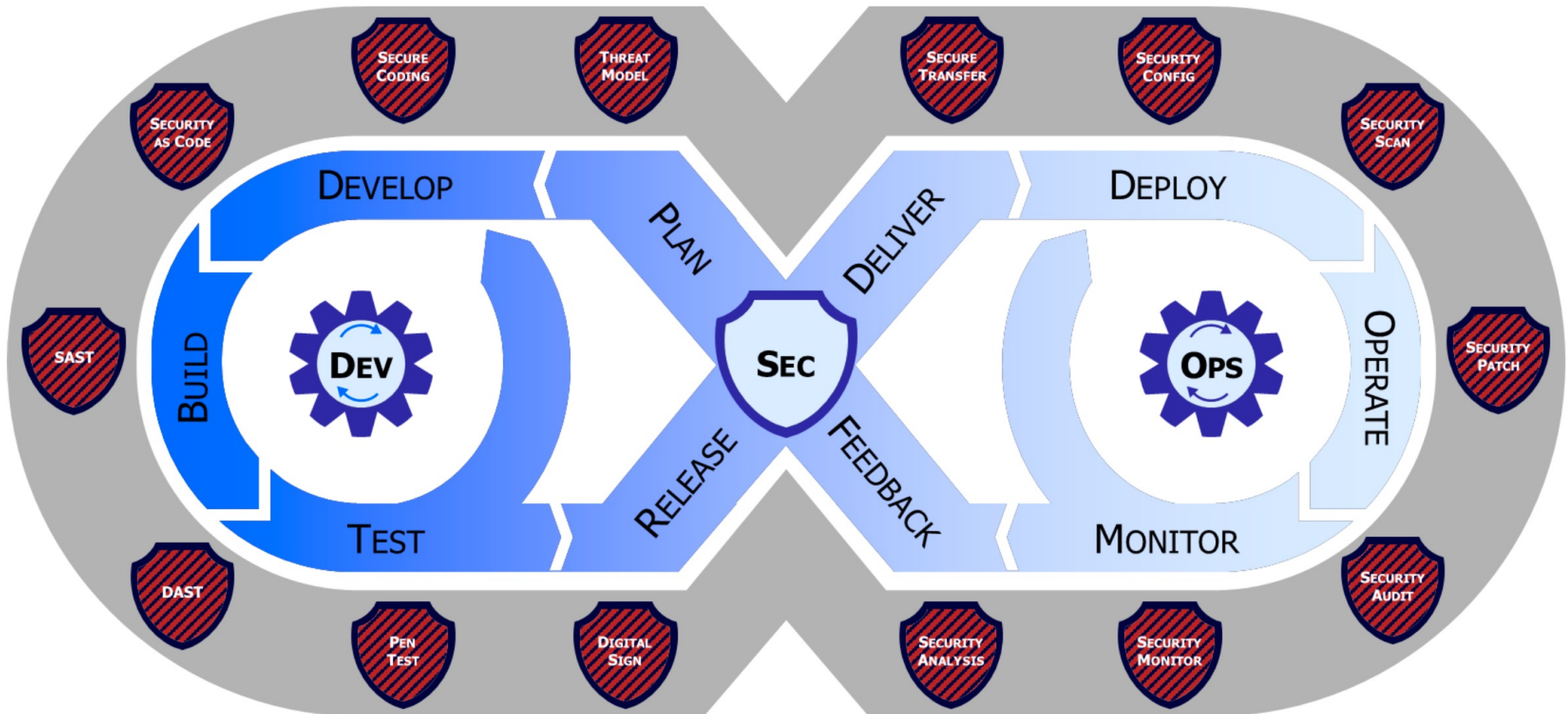


DevSecOps

- ¿Qué es DevSecOps?
 - Es la **integración proactiva** de la **seguridad** en el flujo de trabajo de **DevOps** de modo que las actividades de seguridad sean componentes intrínsecos (no fases separadas) del ciclo de vida del software.
- Principios clave:
 - **"Shift Left"**: Incluir seguridad desde el diseño y desarrollo temprano.
 - **Automatización**: Escanear código, dependencias y configuraciones en cada etapa.
 - **Colaboración**: Equipos de desarrollo, operaciones y seguridad trabajando en conjunto.
- DevSecOps y CI/CD (Integración/Entrega Continua):
 - CI: Automatiza la construcción y pruebas del código.
 - CD: Automatiza el despliegue en entornos de prueba/producción.

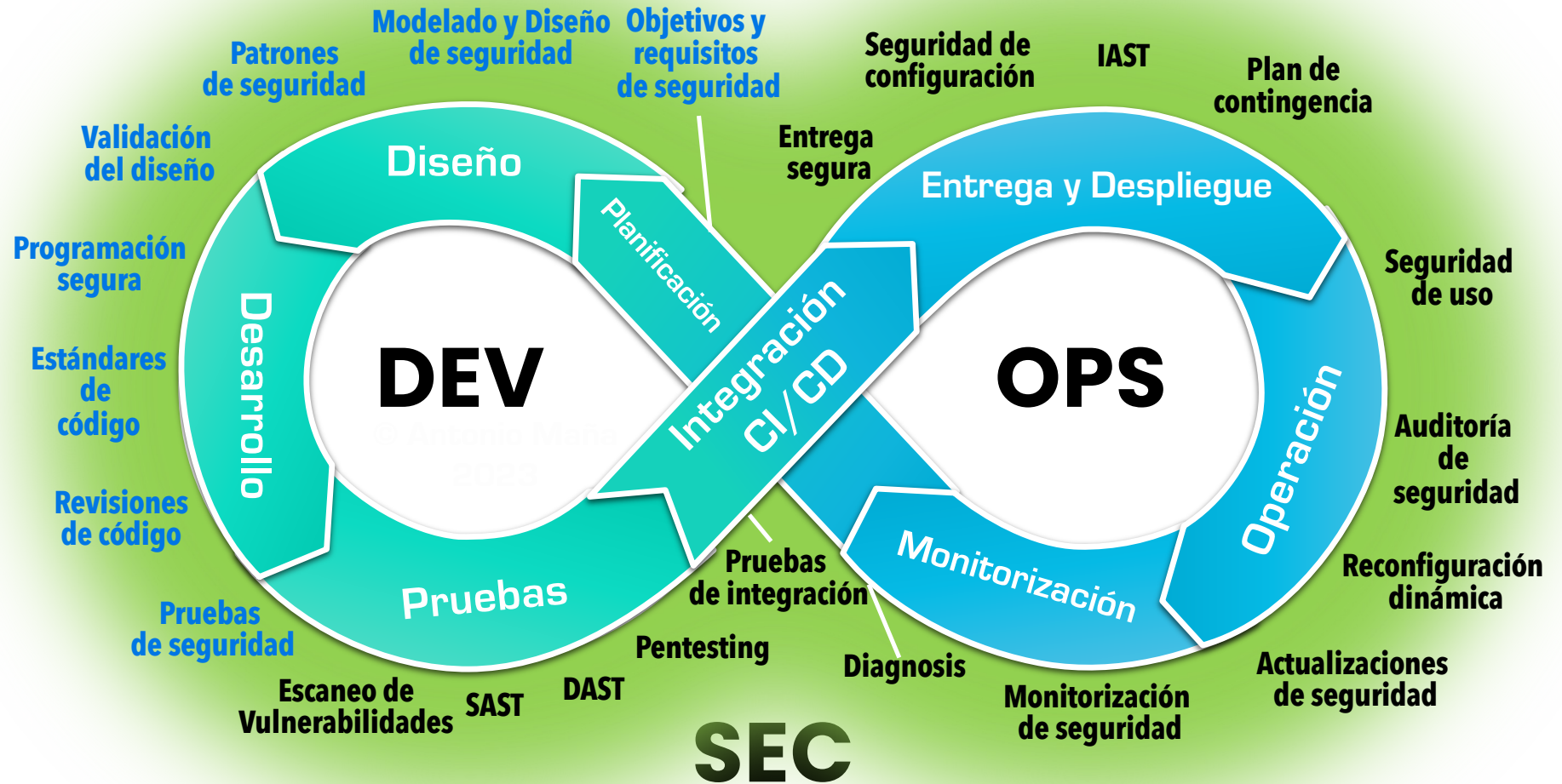


SEGURIDAD EN EL CICLO DE VIDA (DevSecOps DOD)





SEGURIDAD EN EL CICLO DE VIDA (DevSecOps UMA)





HERRAMIENTAS DevSecOps POPULARES

- SAST:
 - SonarQube, Checkmarx.
- SCA:
 - Snyk, Dependabot.
- DAST:
 - OWASP ZAP, Nessus.
- Gestión de secretos:
 - HashiCorp Vault, AWS Secrets Manager.
- Infraestructura como código (IaC):
 - Terraform + Terrascan.
- Referencias:
 - <https://medium.com/@rahulsharan512/devsecops-using-github-actions-building-secure-ci-cd-pipelines-5b6d59acab32>
 - <https://medium.com/cloud-native-daily/building-a-devsecops-pipeline-with-open-source-tools-ad4fd0e13515>



EJEMPLO SIMPLE DE FUNCIONAMIENTO

- **Problema:** Una app web con vulnerabilidad XSS (CVE-2023-12345).
- **Proceso DevSecOps:**
 1. SAST detecta el fallo durante el *pull request*.
 2. El desarrollador corrige el código.
 3. El flujo CI/CD bloquea el despliegue hasta que se resuelve.
- **Resultado:** Vulnerabilidad mitigada en minutos, que no llega a producción.



BENEFICIOS DevSecOps

- Velocidad y seguridad:
 - Detección temprana de vulnerabilidades (ahorra un 80% de costes vs. post-producción).
- Cumplimiento:
 - Automatización de auditorías (ej.: GDPR, HIPAA).
- Resiliencia:
 - Menor riesgo de brechas por supply chain attacks.
- Referencia:
 - <https://github.com/resources/articles/devops/devsecops>



CONCLUSIONES DevOps Y DevSecOps

- DevOps no es un proceso como tal sino una forma de organizar el desarrollo y la operación como un flujo continuo.
 - DevSecOps no es "DevOps + unos escaneres de seguridad", sino una cultura de seguridad continua que busca proteger sin frenar la innovación.
 - DevOps y DevSecOps no sustituyen a otras metodologías de análisis, diseño, modelado, etc., sino que las integran en el ciclo de desarrollo
-
- Referencias y estándares
 - OWASP DevSecOps Maturity Model: Guía para implementación gradual.
 - NIST SP 800-204: Seguridad en sistemas CI/CD.
 - BSIMM: Modelo de madurez de construcción de seguridad.



PREGUNTAS