Universidad del Rosario | Escuela de Ingeniería, Ciencia y Tecnología | MACC Matemáticas Aplicadas y Ciencias de la Computación | HINNT Hub de INNovación y Transferencia

Seguridad en el Ciclo de Desarrollo de Software

Daniel Díaz-López

Líder de Ciberseguridad - MACC
Profesor principal de carrera

danielo.diaz@urosario.edu.co

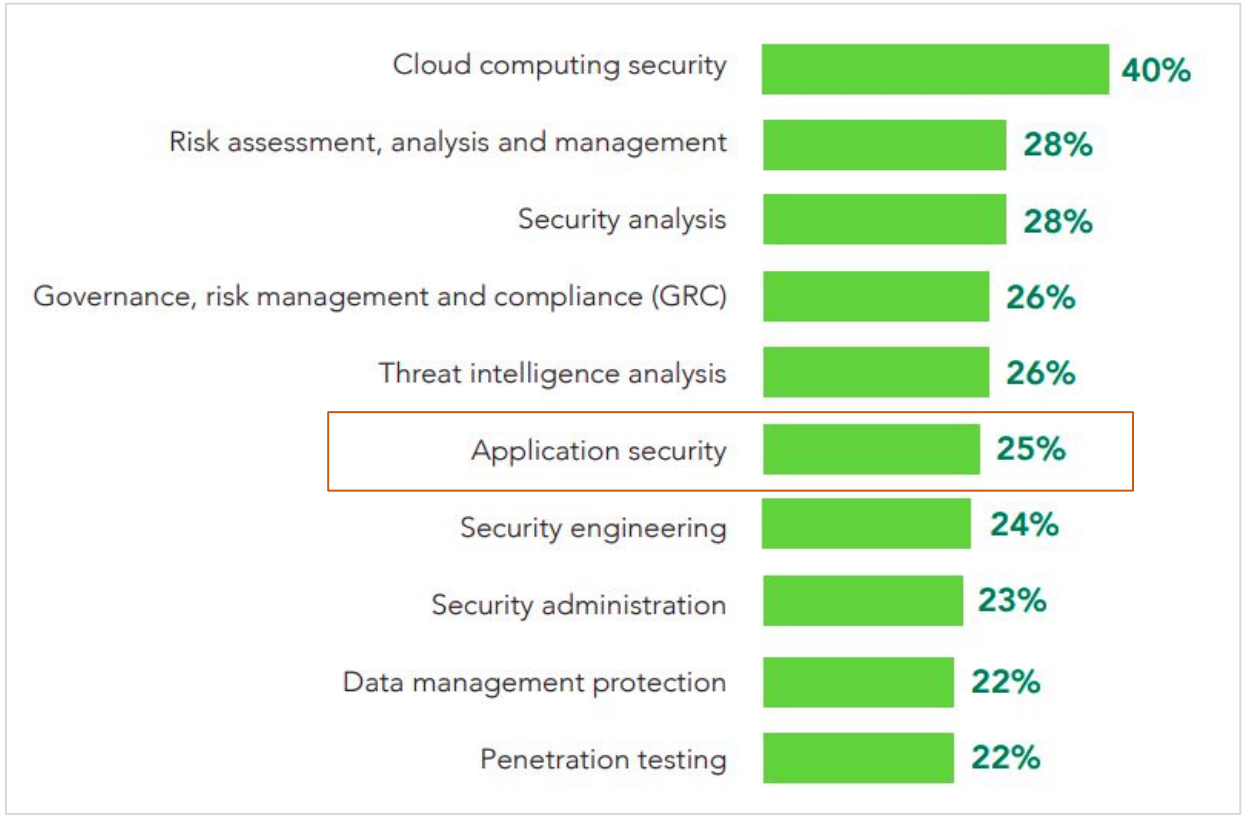
@MACC_URosario @MACC.URosario macc_ur

Agenda

i. ¿Porque seguridad para aplicaciones?
ii. Seguridad holística
iii. Conceptos de software seguro
iv. Tipos de requerimientos de seguridad
v. Caso de uso

¿Por qué seguridad para aplicaciones?

Q: Habilidades en ciberseguridad más demandadas

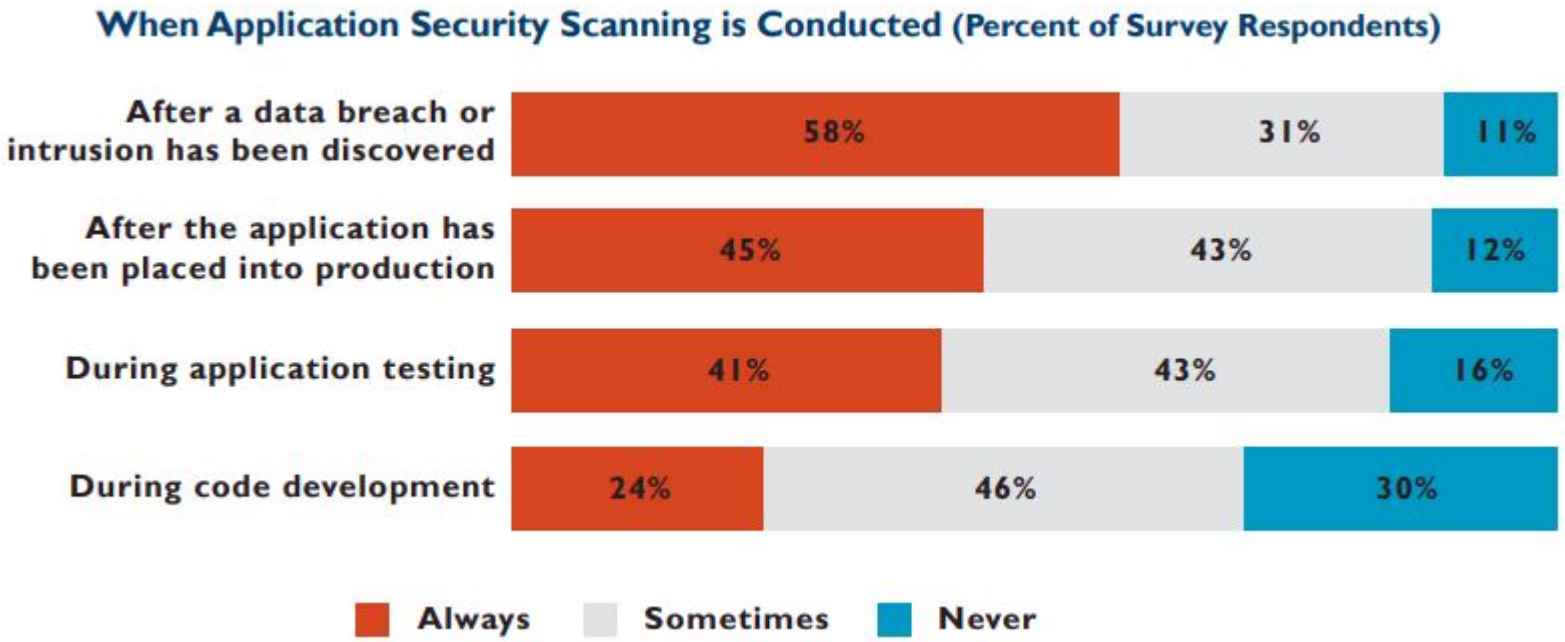


Fuente: "Cybersecurity Professionals Stand Up to a Pandemic",
(ISC)2 CYBERSECURITY WORKFORCE STUDY, 2020, pag 36

Las empresas requieren actualmente personal capacitado en seguridad de aplicaciones

¿Por qué seguridad para aplicaciones?

Q: ¿Cuándo se realiza el escaneo de seguridad de la aplicación (porcentaje de encuestados)?

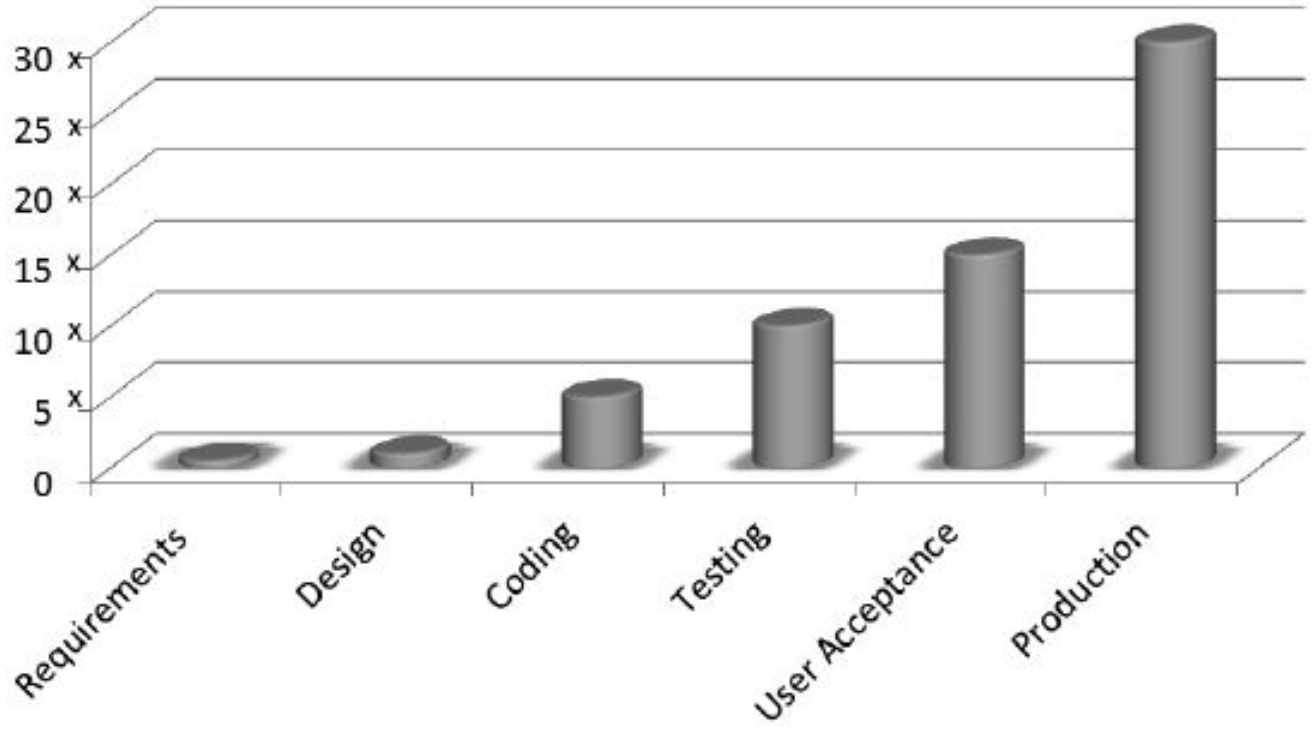


Conclusión:
En una mayor proporción las revisiones de seguridad de aplicaciones ocurren después de un incidente

Fuente: “The 2015 (ISC)2 Global Information Security Workforce Study”, Frost & Sullivan White Paper.

¿Por qué seguridad para aplicaciones?

Q: Costo de solucionar problemas de código en diferentes etapas del ciclo de desarrollo

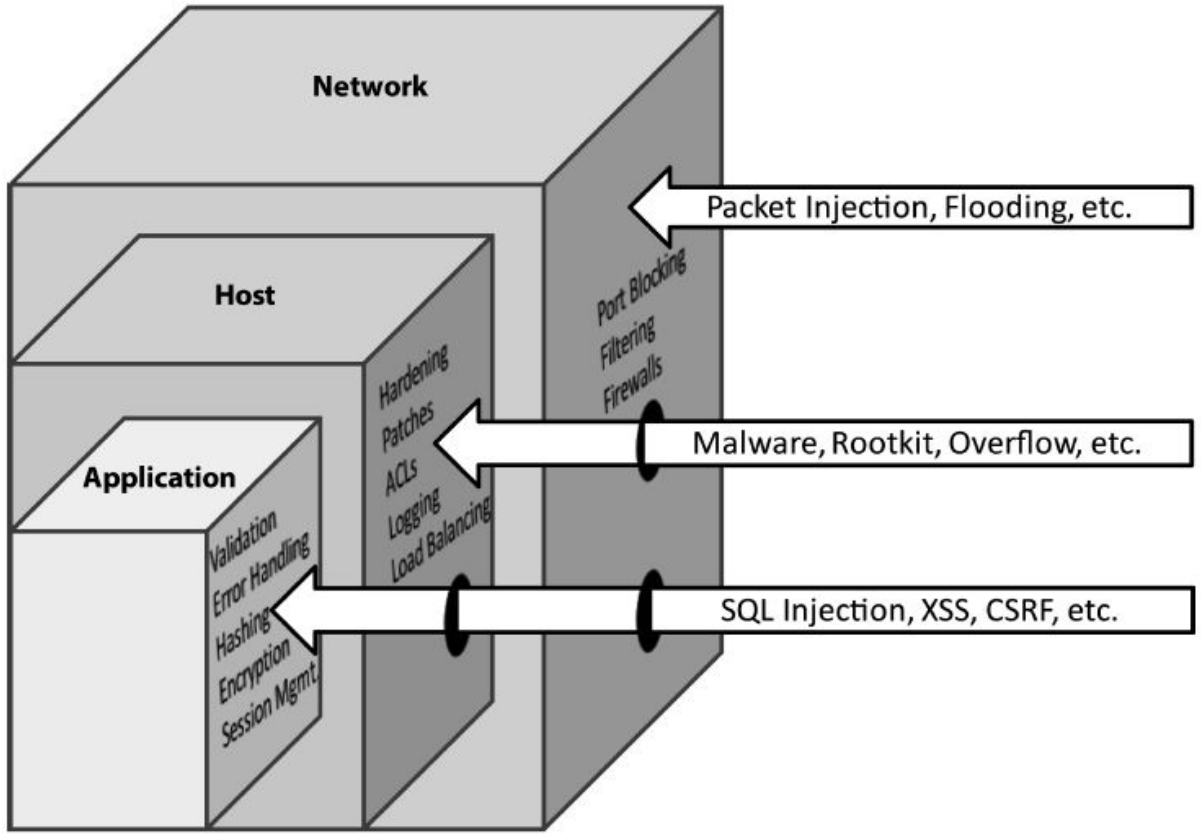


Conclusión:
El costo de mitigar un problema aumenta a medida que se avanza en el ciclo de desarrollo

Fuente: OFFICIAL (ISC) GUIDE TO THE CSSLP CBK SECOND EDITION, Edited by Mano Paul - CSSLP, CISSP, ISC2
https://www.routledge.com/Official-ISC2-Guide-to-the-CSSLP-CBK/Paul/p/book/9781466571273

Seguridad holística

Significa aplicaciones seguras que se ejecutan en hosts seguros (SO) en redes seguras. Es fundamental reconocer que el software es tan seguro como el eslabón más débil.

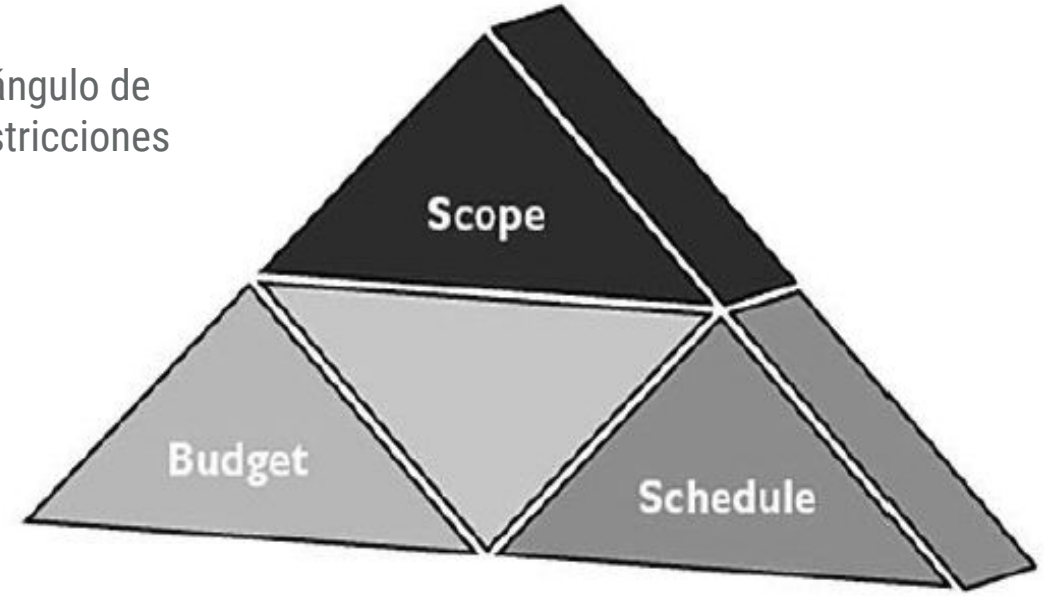


Fuente: OFFICIAL (ISC) GUIDE TO THE CSSLP CBK SECOND EDITION, Edited by Mano Paul, ISC2, CRC Press, Taylor & Francis Group, 2014

Seguridad holística

¿Por qué la seguridad integral no se puede lograr de forma regular?
Los proyectos de desarrollo de software siempre se definen por alcance, presupuesto y cronograma, y casi siempre son rígidos y sin espacio para la seguridad.

Triángulo de Restricciones



Fuente: OFFICIAL (ISC) GUIDE TO THE CSSLP CBK SECOND EDITION, Edited by Mano Paul, ISC2, CRC Press, Taylor & Francis Group, 2014.

Muchos gerentes de proyectos de desarrollo de software no consideran que las inversiones en seguridad tengan un retorno, lo que resulta en:

Security como un “add-on”

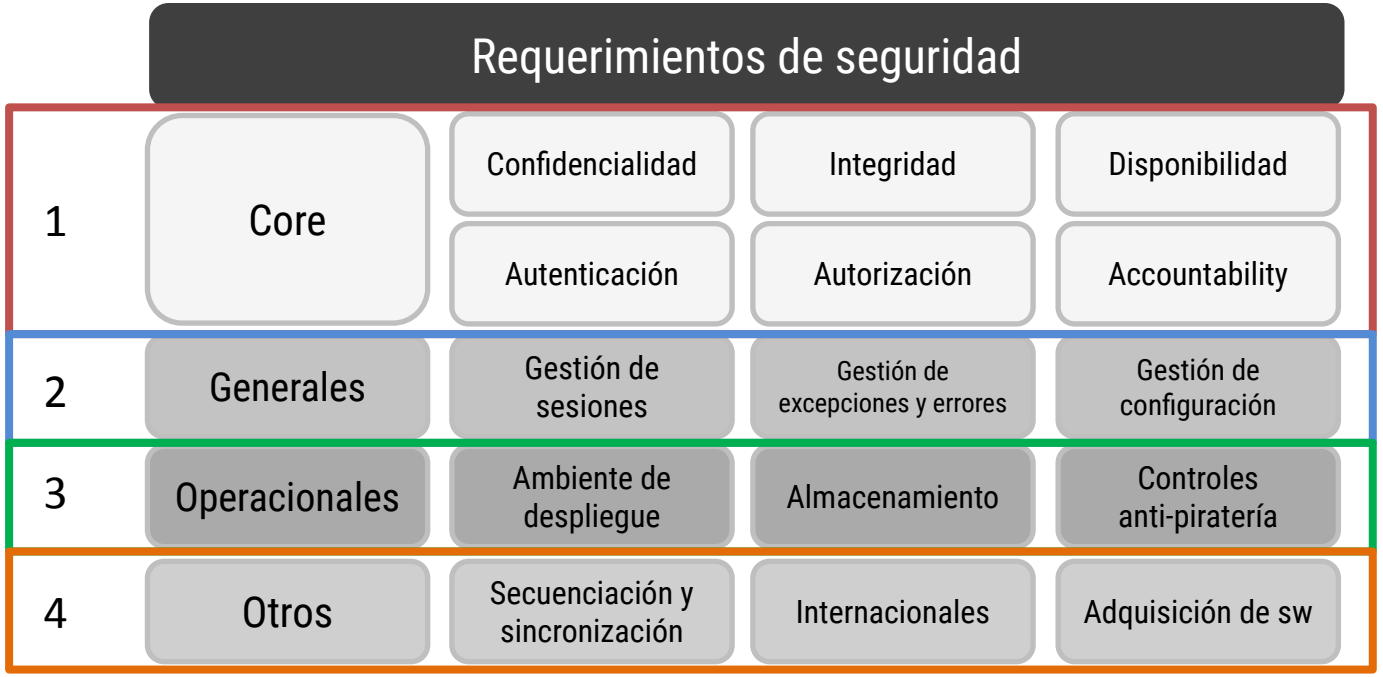
Lo cual es una mala práctica

Conceptos de Software Seguro

Fuente: Imagen adaptada de OFFICIAL (ISC) GUIDE TO THE CSSLP CBK SECOND EDITION, Edited by Mano Paul, ISC2, CRC Press, Taylor & Francis Group, 2014

Tipos de requerimientos de seguridad

- Requerimientos internos (organizational, end user business functionality) y externos (regulations, compliance)
- Para cada concepto de seguridad [core y de diseño] hay un requerimiento.

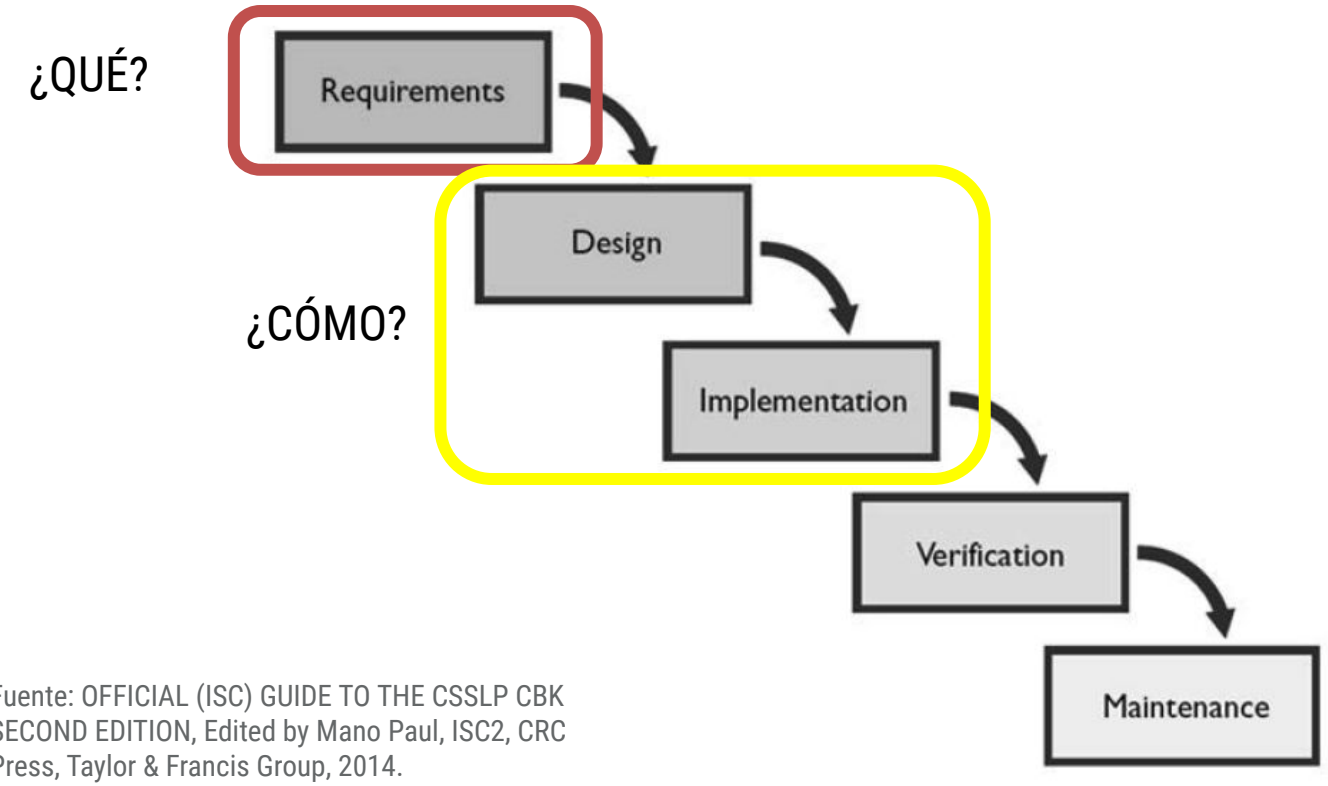


Fuente: Imagen adaptada de OFFICIAL (ISC) GUIDE TO THE CSSLP CBK SECOND EDITION, Edited by Mano Paul, ISC2, CRC Press, Taylor & Francis Group, 2014.

Esencial para obtener requerimientos de seguridad: Traducir especificaciones funcionales en requerimientos de seguridad

Tipos de requerimientos de seguridad

- En la fase de requerimientos definimos todos los tipos de requerimientos (Con lo que debe cumplir el software).
- En las fases de diseño e implementación definimos como implementar dichos requerimientos.



Fuente: OFFICIAL (ISC) GUIDE TO THE CSSLP CBK SECOND EDITION, Edited by Mano Paul, ISC2, CRC Press, Taylor & Francis Group, 2014.

Tipos de requerimientos de seguridad

Requerimientos de Confidencialidad

Ejemplos:

- Cuando el usuario ingresa un PIN en un portal de pago, el PIN debe estar enmascarado con asteriscos.
- Las credenciales deben ser comunicadas al usuario en un manera segura que evite que más de una persona las conozca.
- Toda la información clasificada como top secret debe ser almacenada en el servidor de archivos usando un algoritmo de cifrado validado.
- El administrador del sistema no debe conocer los passwords de ninguna forma. El valor del hash asociado al password puede ser almacenado si un algoritmo de hash seguro es utilizado.
- La comunicación entre el servidor y el cliente debe ser protegida adecuadamente para evitar un ataque de hombre en el medio.

Tipos de requerimientos de seguridad

Requerimientos de Integridad

Aborda 2 áreas:

- Confiabilidad: Integridad del sistema. Garantía de que el software se comporta como se espera
- Precisión: Integridad de datos

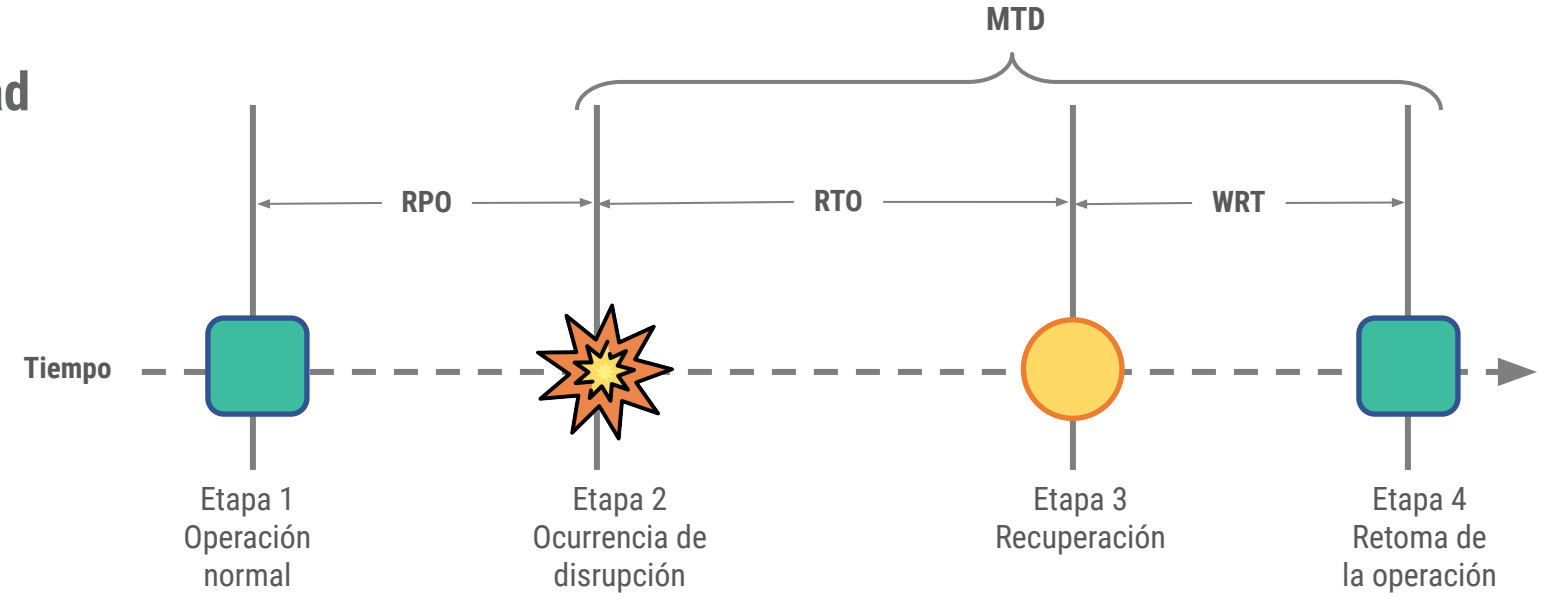
Ejemplos:

- El nombre del archivo debe contener caracteres regulares y números. Otros caracteres deben ser prohibidos.
- Las descargas deben mostrar el valor del hash de tal forma que el usuario pueda validar la integridad de los archivos.
- Los logs deben ser protegidos contra modificación desde el momento en que son generados.
- En un sistema smartgrid, los datos provenientes de un nodo no pueden ser aceptados si estos no han sido validados en cuanto a completitud e integridad.
- Los registros clínicos pueden ser únicamente modificados por las miembros del grupo de usuarios “personal médico”.

Tipos de requerimientos de seguridad

Requerimientos de Disponibilidad

Evitar que errores del software (punteros nulos, asignación inadecuada de memoria, bucles infinitos) hagan el software indisponible.



- **MTD (Maximum Tolerable Downtime):** Máxima cantidad de tiempo que el software puede estar sin operar.
- **WRT (Work Recovery Time):** Tiempo necesario para validar que el software está operando correctamente después de la recuperación.
- **RTO (Recovery Time Objective):** Cantidad máxima de tiempo para restaurar el software.
- **RPO (Recovery Point Objective):** Cantidad máxima permitida de pérdida de datos en el evento de un error en el software.
- **SLA (Service Level Agreement):** Expresa el MTD y el RTO para el software.

Caso de uso:

A partir de la descripción del escenario que se describe en el documento adjunto, identificar los requerimientos de software más apropiados para garantizar el cumplimento de los objetivos organizacionales.

Universidad del Rosario | Escuela de Ingeniería, Ciencia y Tecnología | MACC Matemáticas Aplicadas y Ciencias de la Computación | HINNT Hub de INNovación y Transferencia

Metodologías de Desarrollo de Software

Daniel Díaz-López

Líder de Ciberseguridad - MACC
Profesor principal de carrera

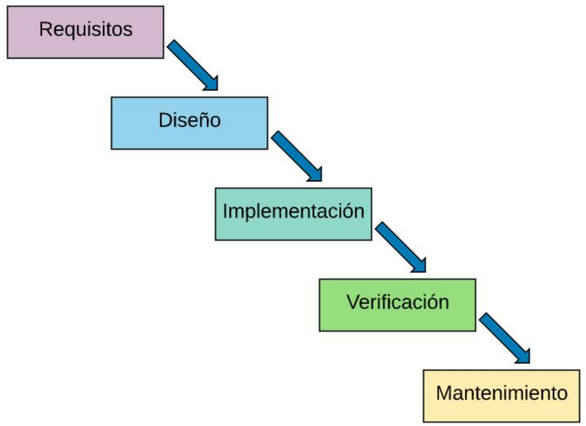
danielo.diaz@urosario.edu.co

@MACC_URosario | @MACC.URosario | macc_ur

1. Metodologías de desarrollo comunes
2. Metodologías de desarrollo AGILE: Scrum
3. Manifiesto por el desarrollo AGILE
4. Principios del desarrollo de software AGILE
5. Roles SCRUM
6. Actividades SCRUM
7. Flujo de desarrollo AGILE: Scrum

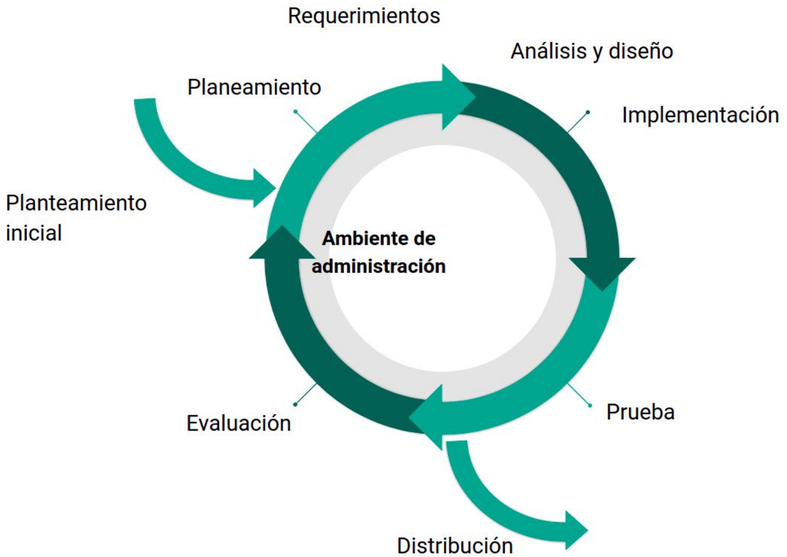
Metodologías de desarrollo comunes

Cascada



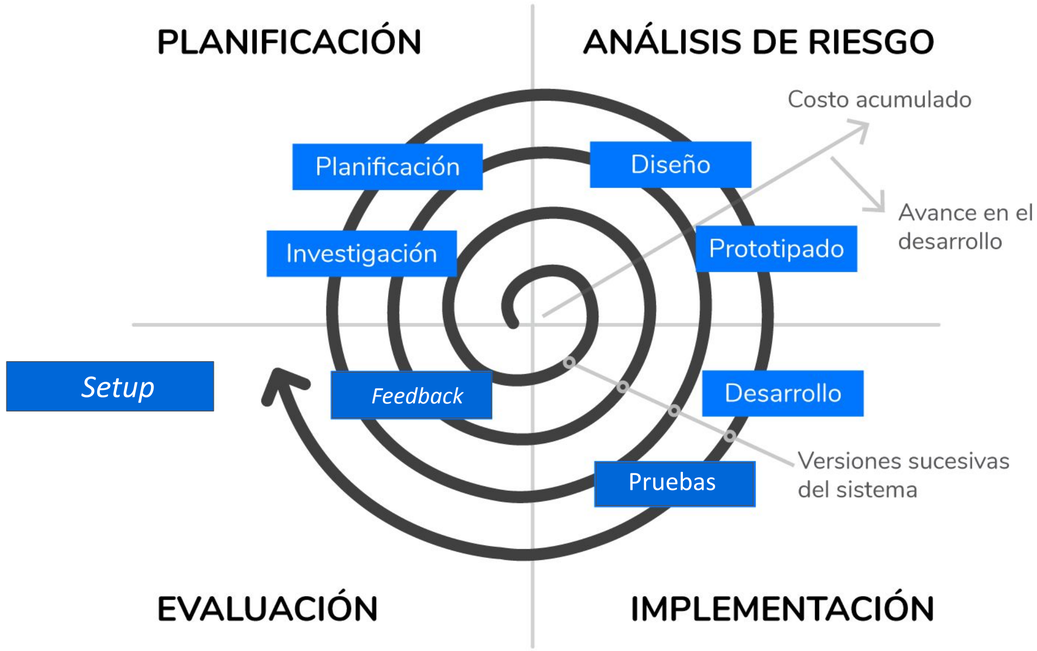
- Estructurado, lineal y secuencial
- Cada fase debe terminar antes de iniciar la siguiente
- Ilustra el sentido en el que fluye el agua (una sola dirección)
- No hay forma de volver atrás

Iterativo



- Desarrollo de pequeños entregables/prototipos
- Desarrollo incremental
- Permite descubrir malos entendidos de manera temprana
- Evita suposiciones

Espiral

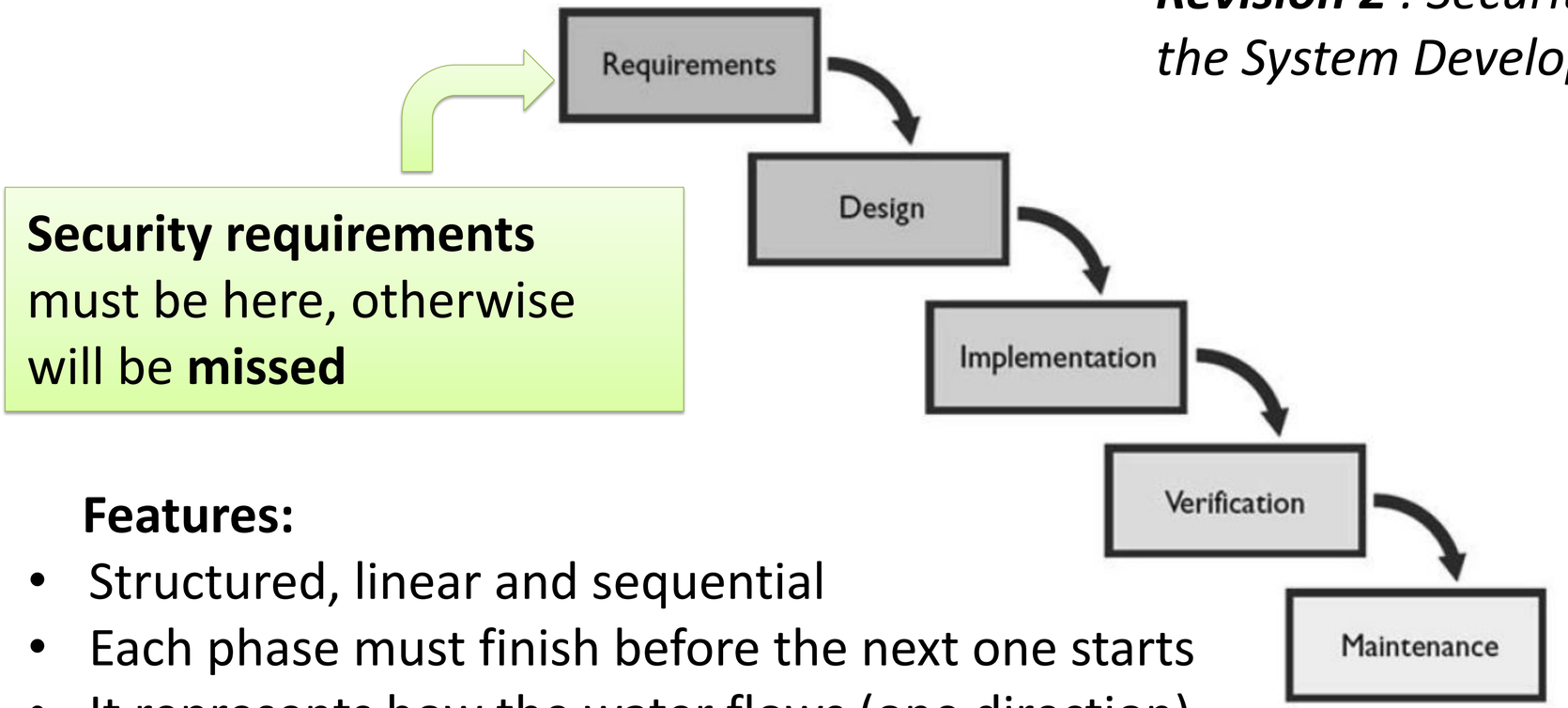


- Espiral = Cascada + Iterativo
- En cada revisión de evaluación de riesgo se decide si el proyecto continúa o no
- Dirigido por la planeación

Software development methodologies

I. Waterfall Model

- Used in ***NIST Special Publication 800-64 Revision 2*** *: Security Considerations in the System Development Life Cycle*



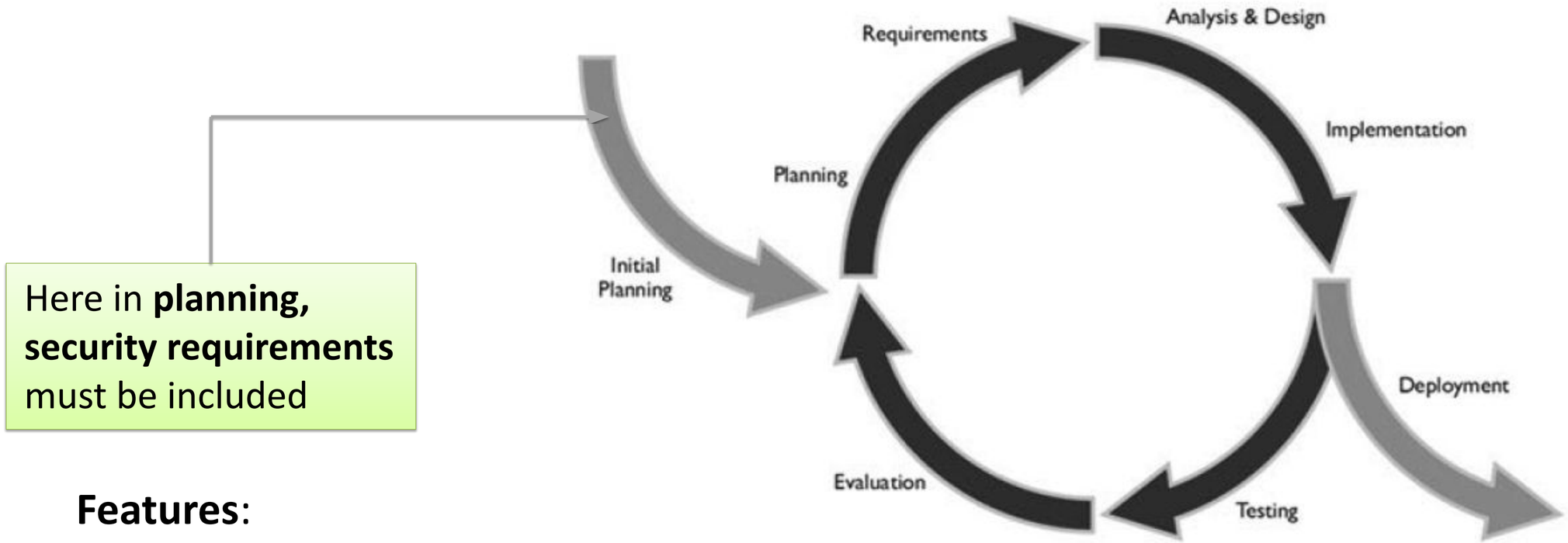
Security requirements must be here, otherwise will be **missed**

Features:

- Structured, linear and sequential
- Each phase must finish before the next one starts
- It represents how the water flows (one direction)
- No way to go back

Software development methodologies

II. Iterative Model



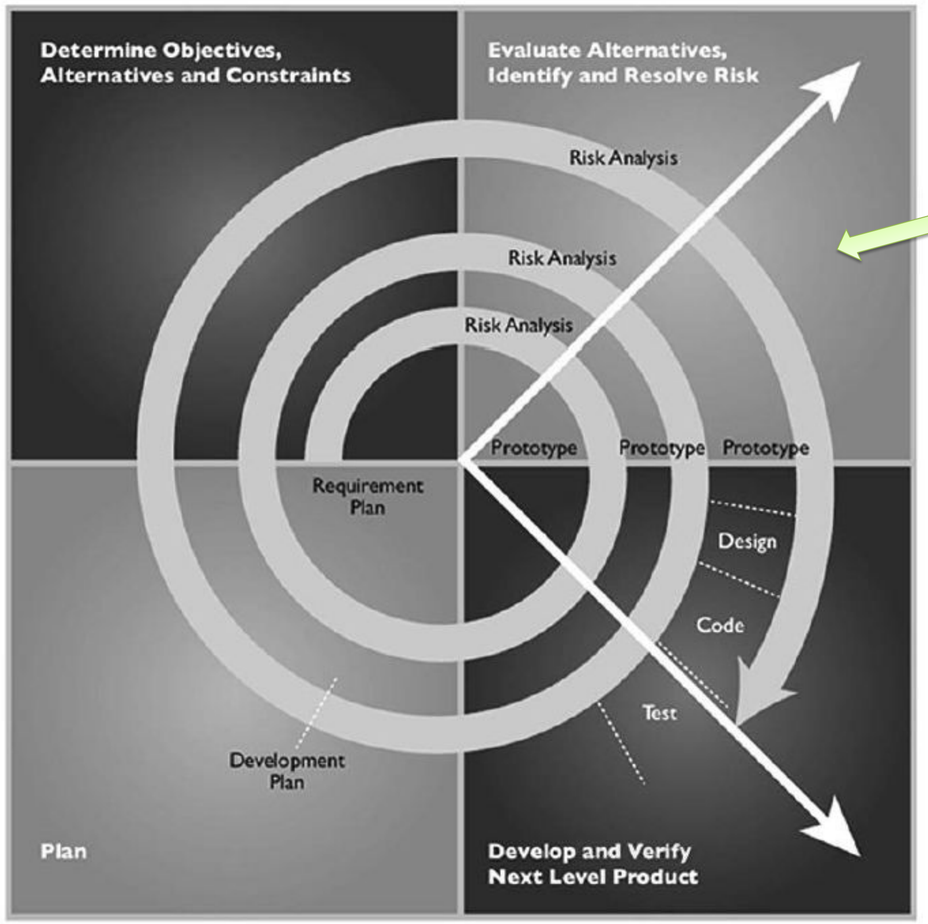
Here in **planning, security requirements** must be included

Features:

- Small deliverables/prototypes
- Incremental development
- It allows to discover misunderstandings early
- It allows avoid assumptions

Software development methodologies

III. Spiral Model



Each phase has a **risk assessment review** activity

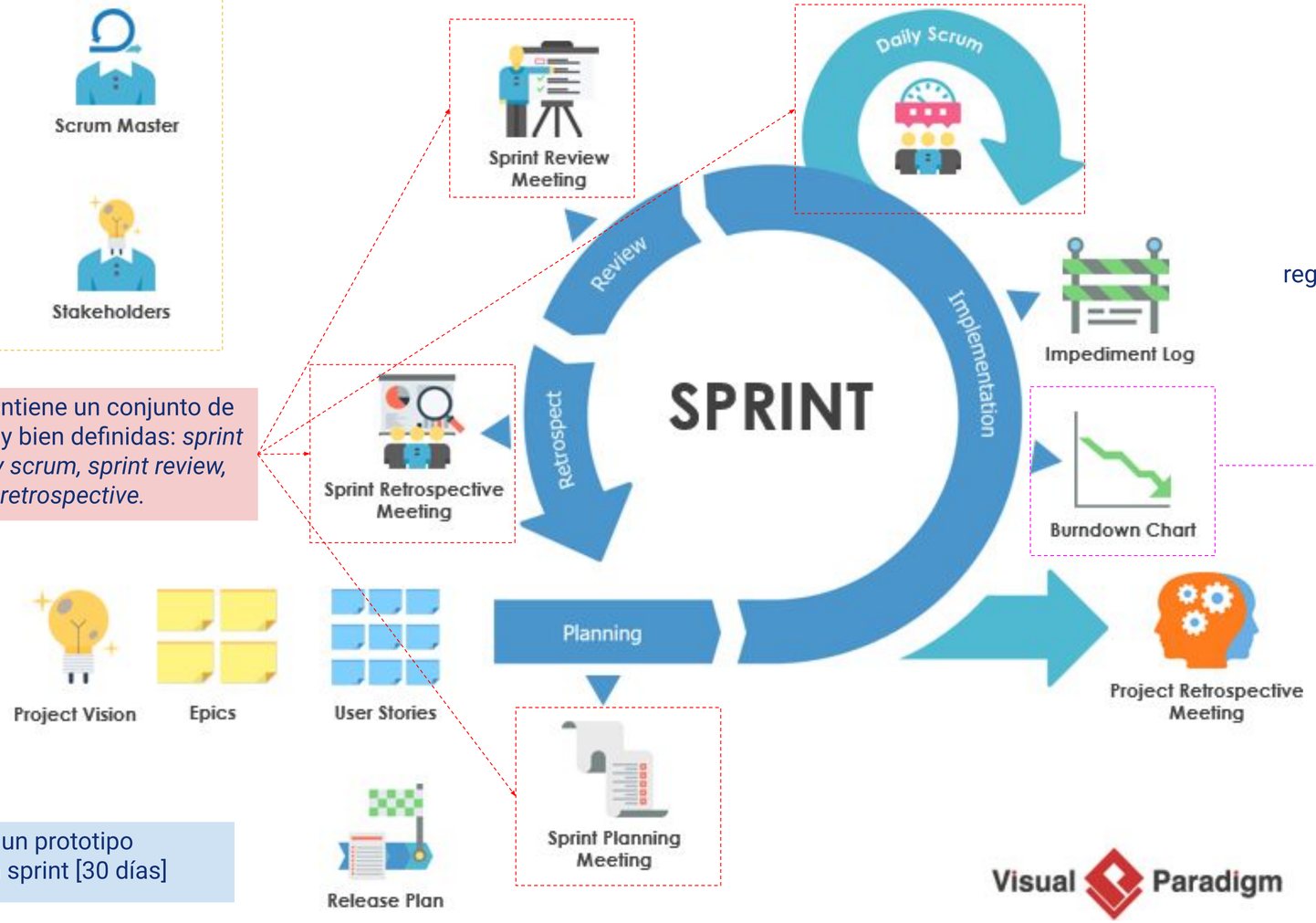
Features:

- Spiral = Waterfall + Iterative
- In the risk assessment review, it is decided if project continues or not
- Driven by planning

Metodología de desarrollo AGILE: Scrum

Involucra un conjunto de roles: equipo de desarrollo, *scrum master*, *product owner*, stakeholders.

Cada *sprint* contiene un conjunto de actividades muy bien definidas: *sprint planning, daily scrum, sprint review, sprint retrospective.*



El progreso diario es registrado en el ***Burn Down Chart*** **(Diagrama de Quemado)**

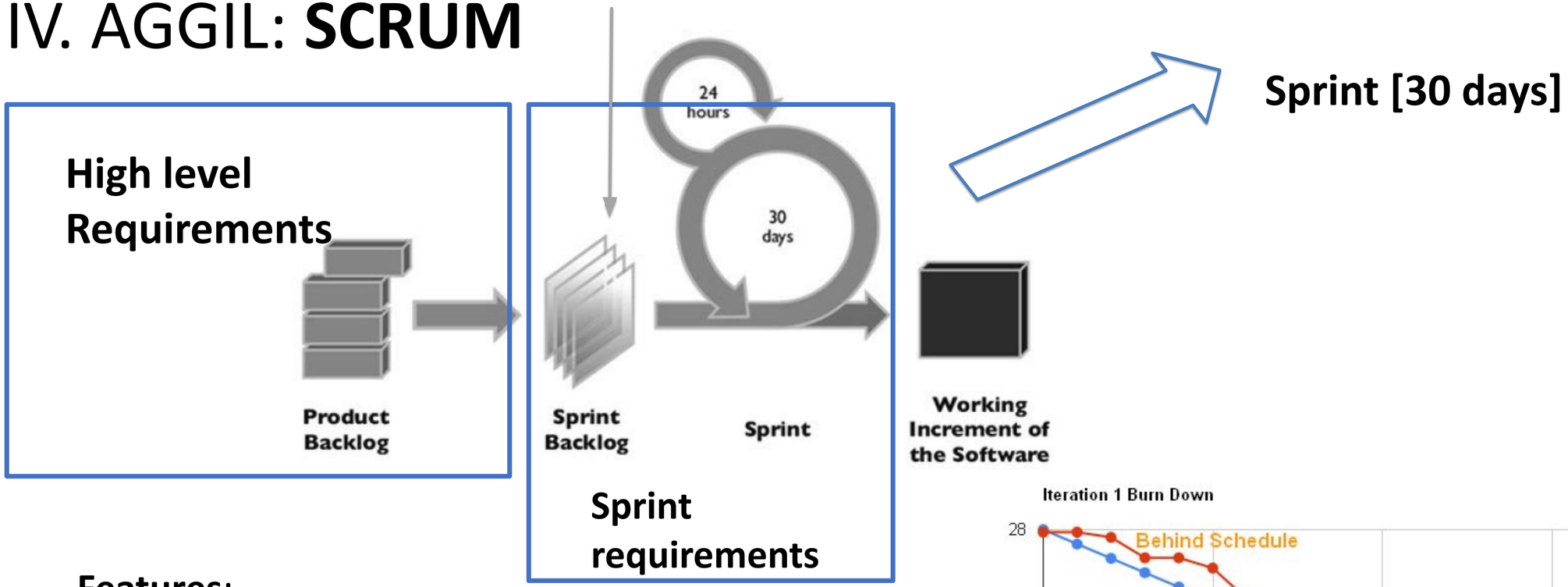
Se debe mostrar un prototipo después de cada sprint [30 días]

Fuente: https://www.visual-paradigm.com/

Software development methodologies

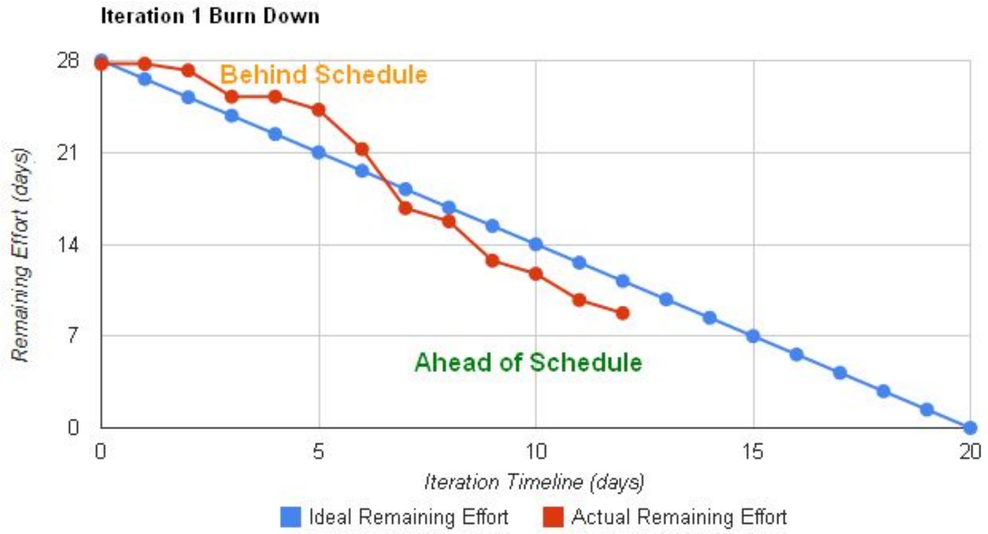
Security requirements must be included here

IV. AGGIL: **SCRUM**



Features:

- Something to show each sprint [30 days]
- Scrum master = Manager
- Product owner = Customer
- Developers
- Team members [5-9]
- Daily progress is recorded in the **Burn Down Chart**



Manifiesto por el desarrollo AGILE

Descubriendo mejores formas de desarrollar software, hemos aprendido a valorar:

Individuos e interacciones

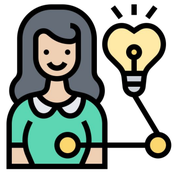
Procesos y herramientas

Software funcionando

Documentación extensiva

Colaboración con el cliente

Negociación contractual



Respuesta ante el cambio

Seguir un plan

Aunque se valoran los elementos de la derecha, se valoran más los de la izquierda

Metodologías de desarrollo AGILE: Scrum - Actividad

Pregunta 1: Dentro de una metodología de desarrollo de software AGILE, se valora más la capacidad de adaptación del equipo ante nuevas circunstancias en lugar del desarrollo estricto de un plan de trabajo predefinido.

- Verdadero
- Falso

Principios del desarrollo de software AGILE

Nuestra mayor prioridad es **satisfacer al cliente** mediante la entrega temprana y continua de software con valor.

Aceptamos que los **requisitos cambien**, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

Entregamos software funcional frecuentemente, entre **dos semanas y dos meses**, con preferencia al periodo de tiempo más corto posible.

Los responsables de negocio y los desarrolladores **trabajamos juntos** de forma cotidiana durante todo el proyecto.

Los proyectos se desarrollan en torno a **individuos motivados**. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación **cara a cara.**

El ***software* funcionando** es la medida principal de progreso.

Los procesos Ágiles promueven el **desarrollo sostenible**. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

La atención continua a la excelencia técnica y al **buen diseño** mejora la Agilidad.

La **simplicidad**, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

Las mejores arquitecturas, requisitos y diseños emergen de equipos **autoorganizados.**

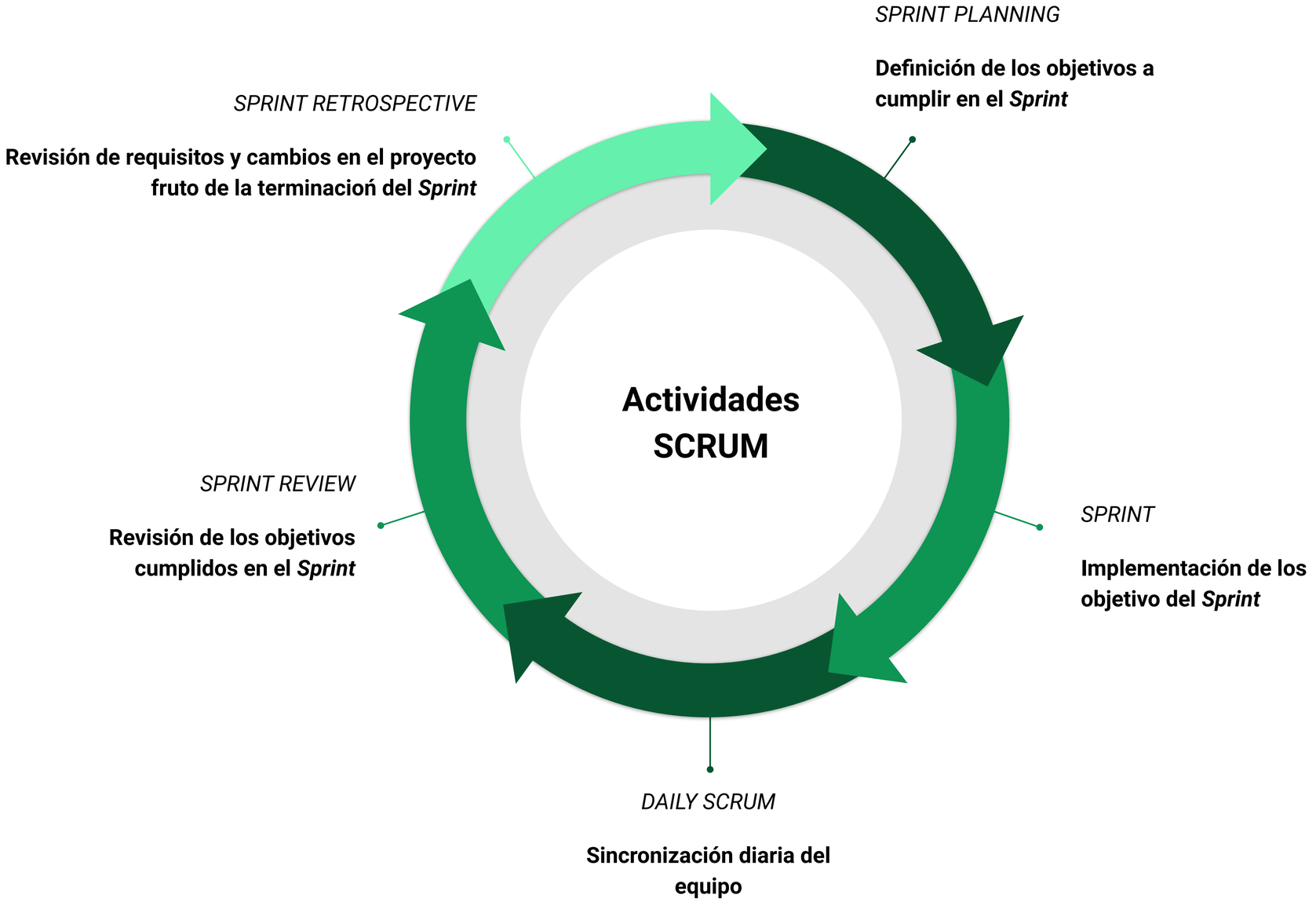
A intervalos regulares el equipo **reflexiona** sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

https://agilemanifesto.org/iso/es/principles.html

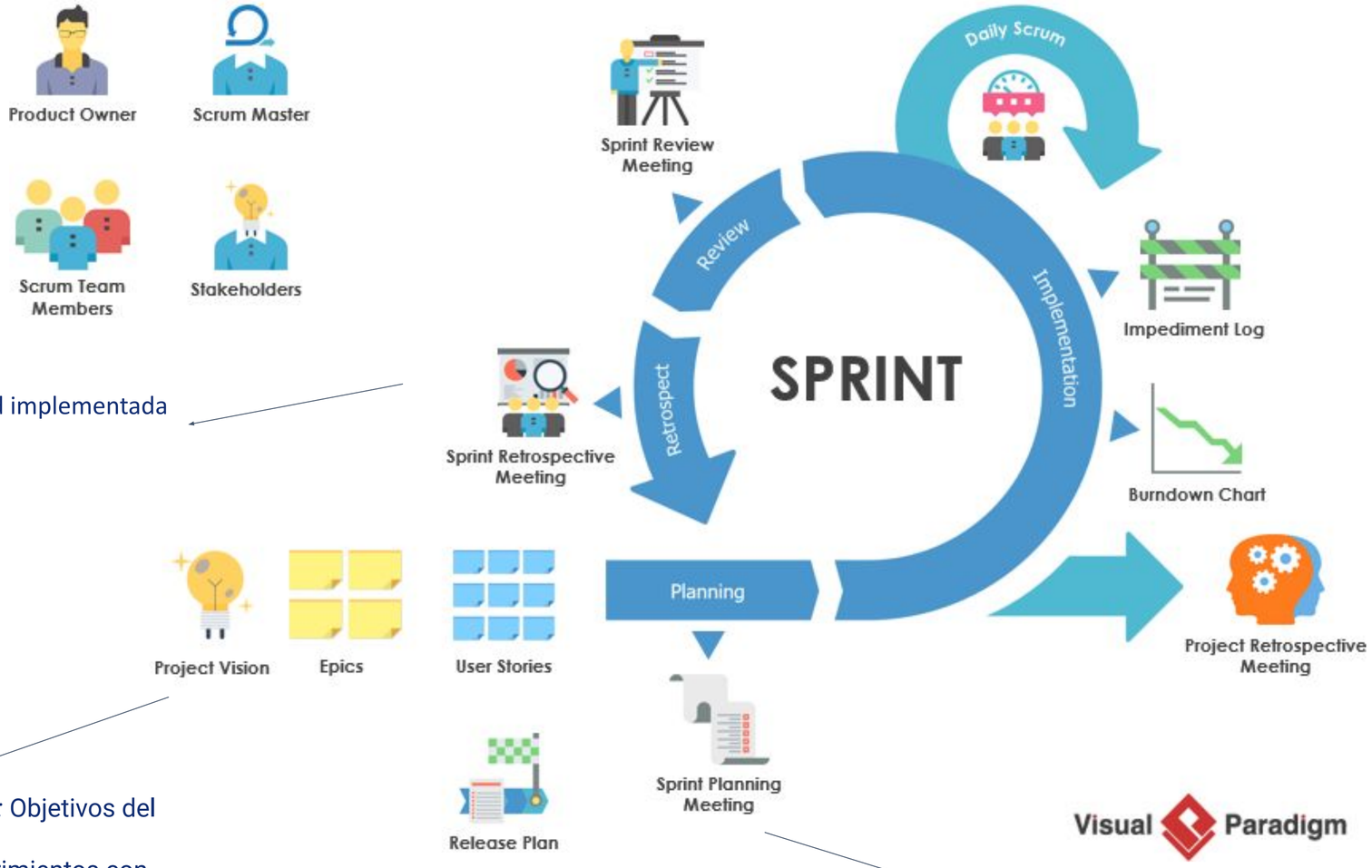
Roles Scrum

01	**Equipo de desarrollo**	• Programadores • *Testers* • Diseñadores, etc.
02	**Dueño del producto - *Product owner***	• Cierra brechas entre el cliente, las partes interesadas del negocio y el equipo de desarrollo. • Experto en el producto y en las necesidades y prioridades del cliente.
03	**Facilitador - *Scrum master***	• Persona responsable de apoyar al equipo de desarrollo • Elimina los obstáculos organizativos y mantiene el proceso agil consistente.
04	**Partes interesadas - *Stakeholders***	• Cualquier persona interesada en el proyecto. • Incluye personas de diferentes departamentos, o incluso de diferentes empresas.
05	**Mentor Agile**	• Tiene experiencia implementando proyectos AGILE • Comparte la experiencia con un equipo de proyecto.

Actividades Scrum



Flujo de desarrollo AGILE: Scrum



- ***Increment:*** Funcionalidad implementada al finalizar cada sprint

- ***Product vision statement***: Objetivos del producto
- ***Product roadmap***: Requerimientos con un alto nivel de especificidad
- ***Product backlog***: Requerimientos con mayor nivel de especificidad

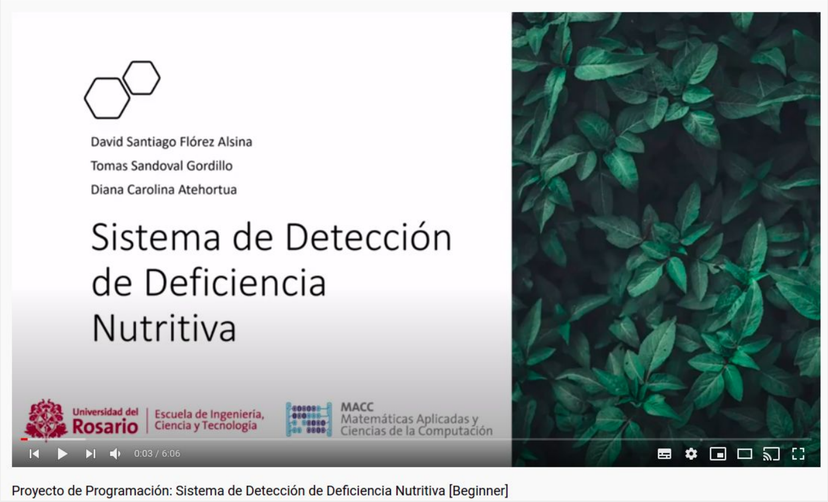
- Tabla de tiempos

- ***Sprint backlog:*** Propósito, historias de usuario y tareas asociadas con un sprint

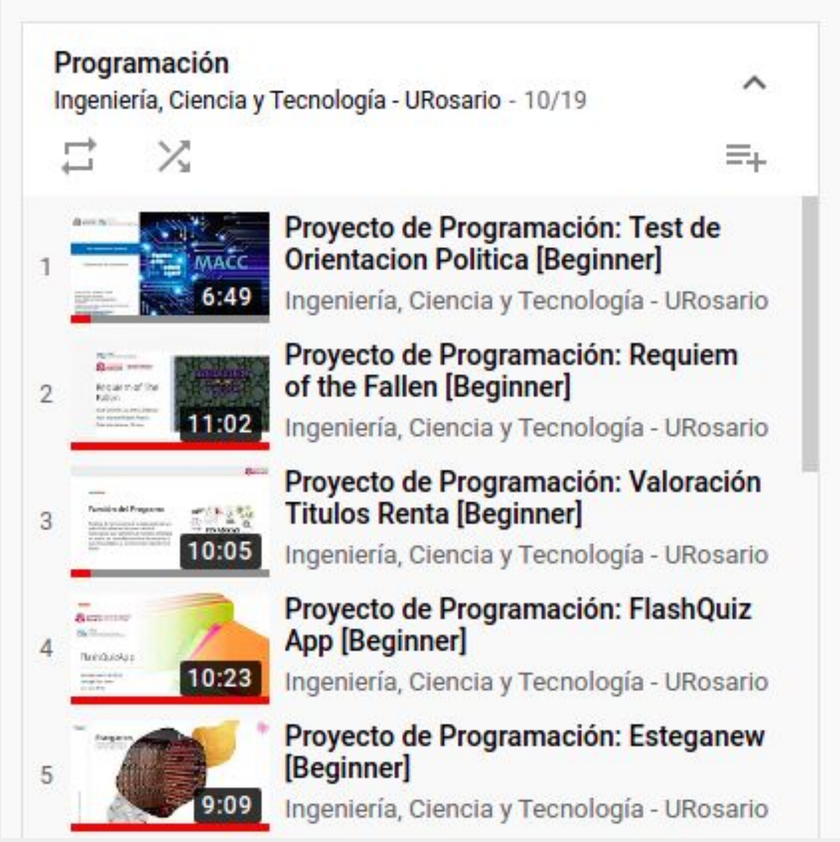
Fuente: https://www.visual-paradigm.com/

¿Quieres aprender mas?

Revisa cómo se desarrollaron algunos proyectos de programación: desde la identificación de una problemática hasta la prueba de las funcionalidades del software



Proyecto de Programación: Sistema de Detección de Deficiencia Nutritiva [Beginner]



Metodologías de desarrollo AGILE: Scrum - Actividad

Pregunta 2: ¿Cual es una ventaja de la metodología de desarrollo de software AGILE? (Seleccione dos respuestas):
Rta correcta: Evita que el equipo de desarrollo se quede inmiscuido en un problema técnico por demasiado tiempo sin visualizar un avance.

- No se maneja un cronograma de trabajo lo que permite alta flexibilidad.
- Permite dividir un gran proyecto en diferentes partes, cada una de ellas abordable en un Sprint.
- Está orientado a la documentación de todo el proceso sin perder detalle.