UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

INGENIERÍA DE REQUERIMIENTOS Y ASEGURAMIENTO DE LA CALIDAD

Sesión 3: Implementación del control de calidad en el desarrollo de software

Contenidos de esta sesión

- Pasos para implementar un proceso efectivo de control de calidad en el desarrollo de software
- Ejemplos de buenas prácticas en la implementación del control de calidad
- Ejercicios prácticos para que los estudiantes puedan aplicar lo aprendido en sus propios proyectos de desarrollo de software

Resultado de Aprendizaje de la Unidad

Aplica diversas normas, modelos y estándares de aseguramiento de la calidad al software y al ciclo de desarrollo, con el fin de asegurar que el software cumpla con los estándares de calidad exigidos en la Organización.

El Problema: Explosión Combinatoria

Pruebas Exhaustivas (2^n)

Cuando tienes una funcionalidad con varias opciones que pueden estar activadas o desactivadas, el número de combinaciones posibles crece de forma **exponencial**. Si quieres probar todas las combinaciones para estar 100% seguro, te enfrentas a la "explosión combinatoria".

Crecimiento Exponencial: 2^n Variables

2 variables	=	4 pruebas	Manejable
5 variables	=	32 pruebas	Empieza a ser molesto
10 variables	=	1.024 pruebas	Poco práctico
20 variables	=	1.048.576 pruebas	Imposible

El Problema Real

Las **pruebas exhaustivas** se vuelven impracticables muy rápidamente. Con solo 20 variables binarias, necesitarías más de **1 millón de casos de prueba**. Esto es una trampa de tiempo y recursos.

La Solución: Pruebas de Pares

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Pairwise Testing - El Enfoque Inteligente

En lugar de probar *todas* las combinaciones, nos enfocamos en probar cada **par posible de valores** al menos una vez. Esto reduce drásticamente el número de casos de prueba manteniendo una cobertura muy alta sobre los errores más probables.

Observación Empírica

La mayoría de los bugs no son causados por la interacción de 3, 4 o 5 variables a la vez

Causa Principal

Los errores suelen ser provocados por una sola variable o por la interacción de un PAR de variables

Estrategia Inteligente

Enfocarse en probar cada par posible de valores al menos una vez

Principio Clave

"La gran mayoría de los bugs son causados por la interacción de un PAR de variables, no por interacciones complejas de múltiples variables."

Resultado: Máxima eficiencia con mínimo esfuerzo

Ejemplo Práctico: De 8 a 4 Pruebas

Supongamos que tenemos 3 variables: **Notificaciones (A)**, **Modo Oscuro (B)** y **Sonidos (C)**. Cada una puede estar Activada o Desactivada.

✕ Pruebas Exhaustivas (2^3 = 8 casos)

Caso	A	B	C
1	Activado	Activado	Activado
2	Activado	Activado	Desactivado
3	Activado	Desactivado	Activado
4	Activado	Desactivado	Desactivado
5	Desactivado	Activado	Activado
6	Desactivado	Activado	Desactivado
7	Desactivado	Desactivado	Activado
8	Desactivado	Desactivado	Desactivado

Pruebas de Pares (4 casos)

Caso	A	B	C
1	Activado	Activado	Activado
2	Activado	Desactivado	Desactivado
3	Desactivado	Activado	Desactivado
4	Desactivado	Desactivado	Activado

8 Casos Exhaustivos → **4** Casos Pairwise **50%** Reducción

¿Por qué funciona?

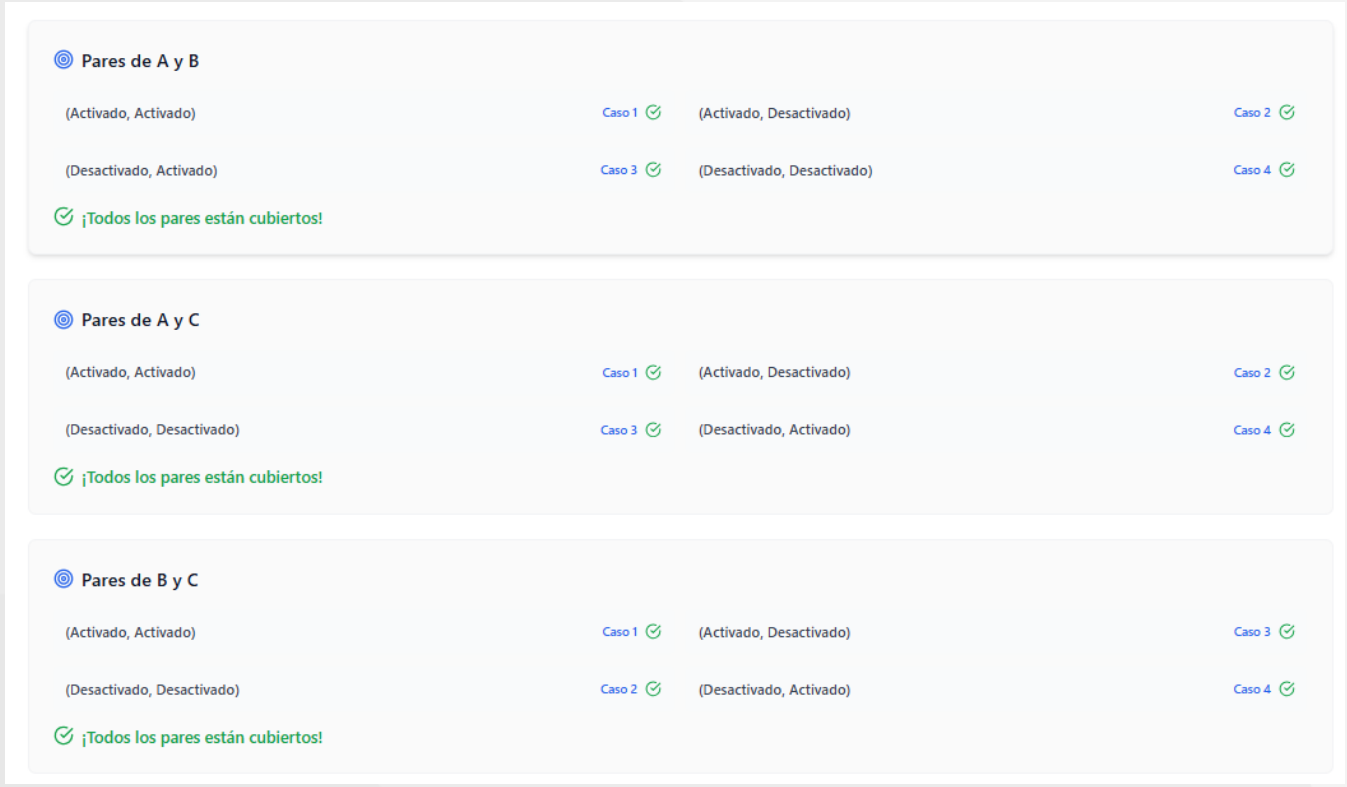
Los 4 casos de pairwise cubren **todos los pares posibles**: (A,B), (A,C) y (B,C). Hemos reducido a la mitad el número de pruebas, y a medida que 'n' crece, el ahorro es mucho mayor.

Análisis de Cobertura

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Verificación de Pares Cubiertos

Analicemos cómo los 4 casos de prueba cubren **todos los pares posibles** de nuestras 3 variables. Cada par de variables debe tener todas sus combinaciones representadas al menos una vez.



3	12	100%
Pares de Variables	Combinaciones de Pares	Cobertura de Pares
(A,B), (A,C), (B,C)	4 combinaciones × 3 pares	Todas cubiertas en 4 casos

Más Allá de lo Binario (V/F)

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

De Variables Binarias a Campos con Múltiples Valores
El ejemplo de Activado/Desactivado es una simplificación. En el mundo real, los campos tienen **múltiples valores**, y es ahí donde se combinan diferentes técnicas de testing para manejar la complejidad.

Ejemplos del Mundo Real
Edad para Crédito
Rango: 18-65 años**Posibilidades:** 48 valores posibles

¿Probar cada edad individual?

Monto del Préstamo

Rango: $1,000 - $100,000**Posibilidades:** 99,000 valores posibles

¿Probar cada monto?

Código Postal

Rango: 00000 - 99999**Posibilidades:** 100,000 valores posibles

¿Probar cada código?

El Desafío Real

¿Qué pasa cuando una variable no es un simple sí/no, sino un campo con un rango, como la **edad para un crédito que se otorga solo a personas entre 18 y 65 años**? Probar cada edad es imposible.

Solución: Antes de pensar en las combinaciones, debemos seleccionar inteligentemente los valores *dentro de cada campo* usando técnicas especializadas.

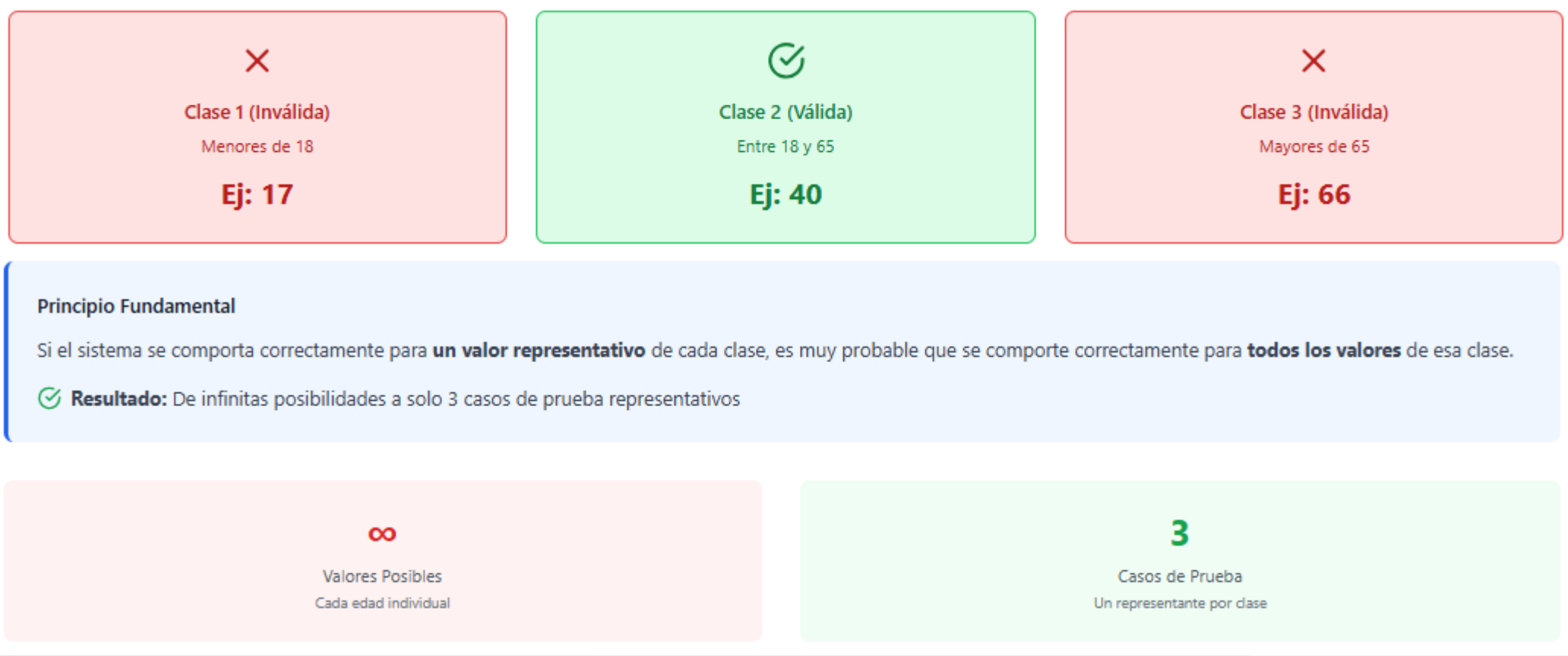
Partición de Equivalencia

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Agrupando los Datos Inteligentemente

En lugar de ver cada edad como un caso único, las agrupamos en **"clases"** donde el sistema debería comportarse de la misma manera. Esta técnica reduce infinitas posibilidades a un conjunto manejable de casos representativos.

Ejemplo: Edad para Crédito (Rango 18-65 años)



Análisis de Valores Límite

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Boundary Value Analysis (BVA) - Donde Viven los Bugs
La experiencia nos dice que los errores no suelen ocurrir con un valor en medio del rango (como la edad 40), sino justo en los **bordes** o **límites**. El BVA se enfoca precisamente en esos puntos críticos.

¿Por qué los límites son críticos?
- Los desarrolladores cometen errores en condiciones de borde (>= vs >)
- Los algoritmos fallan en valores extremos
- Las validaciones suelen tener errores off-by-one
- Los rangos pueden estar mal definidos

Resultado del BVA

Para el campo "Edad", hemos identificado los **6 valores más importantes** a probar:

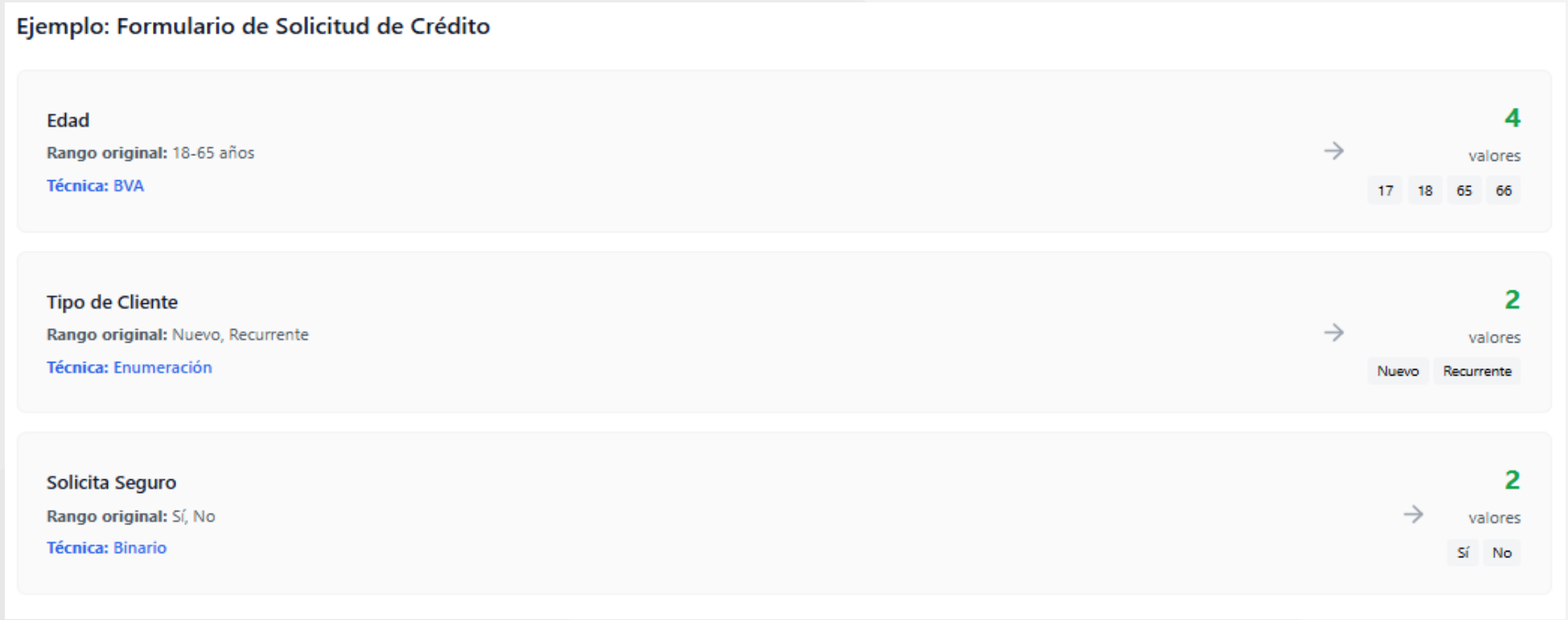
17 18 19 64 65 66

Combinando Técnicas

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

El Flujo de Trabajo Real

Ahora unimos todo. La potencia real viene de combinar diferentes técnicas: primero seleccionamos valores inteligentemente para cada campo, luego aplicamos pairwise testing para las combinaciones.



Combinando Técnicas

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Cálculo de Combinaciones

Pruebas exhaustivas: `4 × 2 × 2 = 16 pruebas` → Con pairwise: ~6-7 pruebas

Resultado: Casos de Prueba Pairwise

Caso	Edad	Tipo Cliente	Seguro
1	17	Nuevo	Sí
2	18	Nuevo	No
3	65	Recurrente	Sí
4	66	Recurrente	No
5	18	Recurrente	Sí
6	65	Nuevo	No

Flujo de Trabajo Integrado

Paso 1: Selección de Valores

Usa **Análisis de Valores Límite** para reducir los posibles valores de cada campo individual a un conjunto pequeño y significativo.

Paso 2: Combinación Inteligente

Alimenta estos conjuntos de valores a una herramienta de **pruebas de pares** para generar el número mínimo de casos de prueba.

Flujo de Trabajo Integrado

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Metodología Completa de Testing Inteligente

La combinación de estas técnicas crea un flujo de trabajo poderoso que maximiza la efectividad del testing mientras minimiza el esfuerzo requerido. Es la aplicación práctica del principio de Pareto en testing de software.

Paso 1

Identificar Campos

Analizar todos los campos de entrada del sistema

Paso 2

Aplicar Técnicas de Selección

Usar Partición de Equivalencia y BVA para cada campo

Paso 3

Generar Combinaciones

Aplicar Pairwise Testing a los valores seleccionados

Beneficios Cuantificables

85-90%	60-80%	70-85%	Alta
Cobertura de Errores	Reducción de Casos	Tiempo de Ejecución	Efectividad
De todos los bugs posibles	Vs. pruebas exhaustivas	Menos tiempo de testing	Encuentra bugs críticos

❌ Enfoque Tradicional

- Pruebas exhaustivas o aleatorias
- Alto número de casos de prueba
- Muchos casos redundantes
- Baja eficiencia tiempo/cobertura
- Difícil de mantener y escalar

✅ Enfoque Inteligente

- Selección científica de valores
- Combinaciones optimizadas
- Máxima cobertura con mínimo esfuerzo
- Enfoque en errores más probables
- Escalable y mantenible

Pasos para implementar un proceso efectivo de control de calidad en el desarrollo de software

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

IDEA
(Concepto, necesidad, problemática)

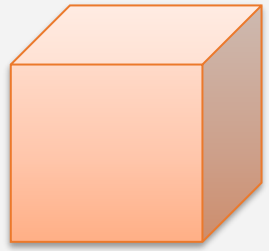
ANÁLISIS

DISEÑO

CONSTRUCCIÓN

IMPLEMENTACIÓN

PRUEBAS



PRODUCTO
(Software, Incremento, MVP, etc.)

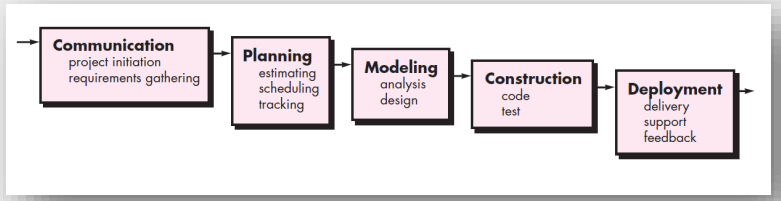
¡Todos los planes, de cualquier tipo, se deben adaptar al modelo de desarrollo de software que defina el proyecto!

Pressman (2002) . Fases de la Ingeniería de Software.

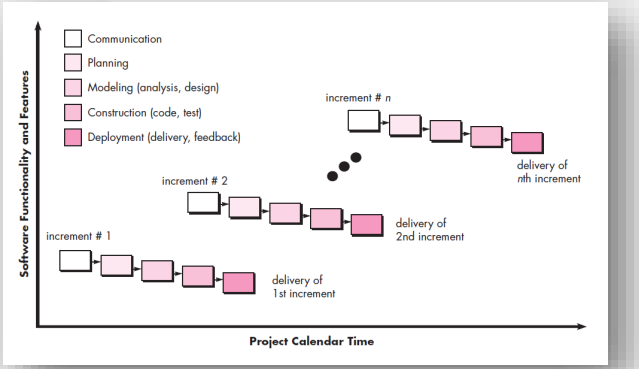
Principales modelos de desarrollo de software

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

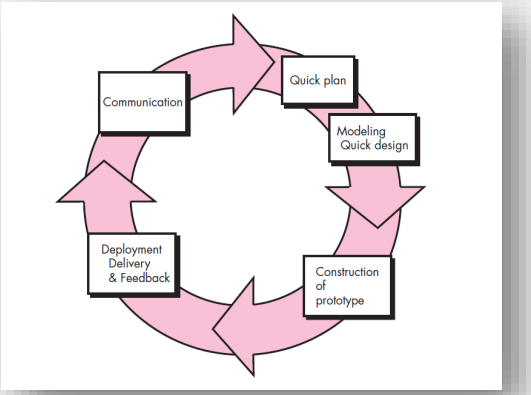
MODELO CASCADA (WATERFALL)



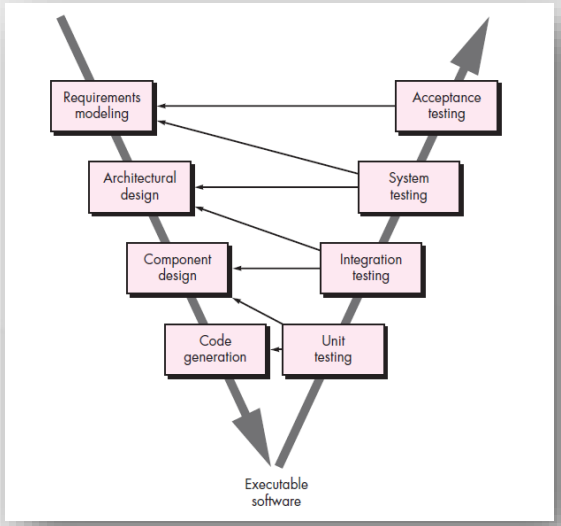
MODELO INCREMENTAL



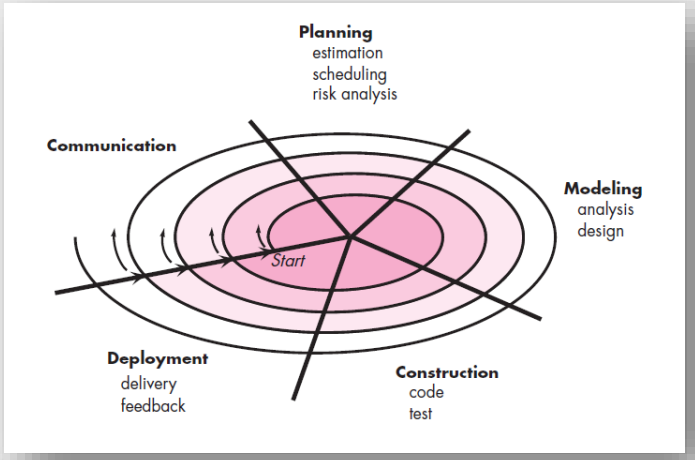
PARADIGMA PROTOTIPADO



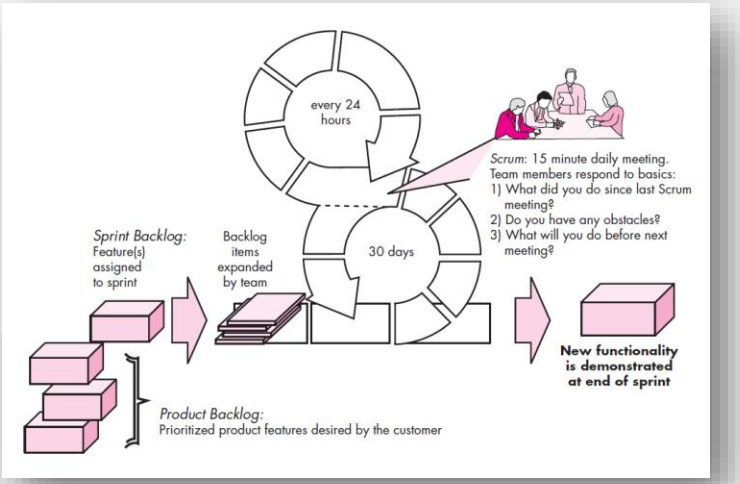
MODELO EN V (V-MODEL)



ESPIRAL DE BOEHM



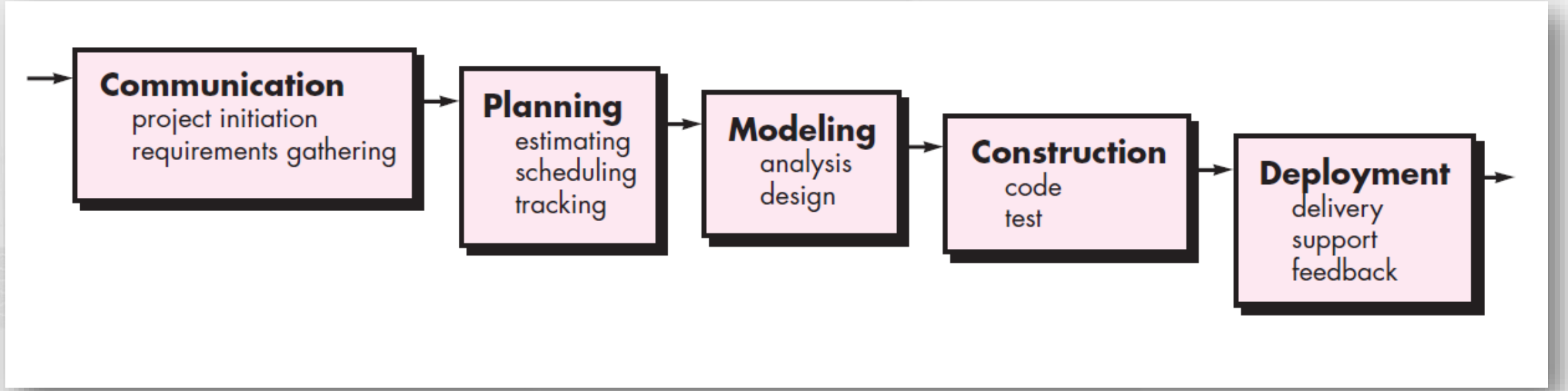
MARCO DE TRABAJO SCRUM



Pressman (2010). Software Engineering 7th Ed. Pp. 39-83.

Modelo Cascada

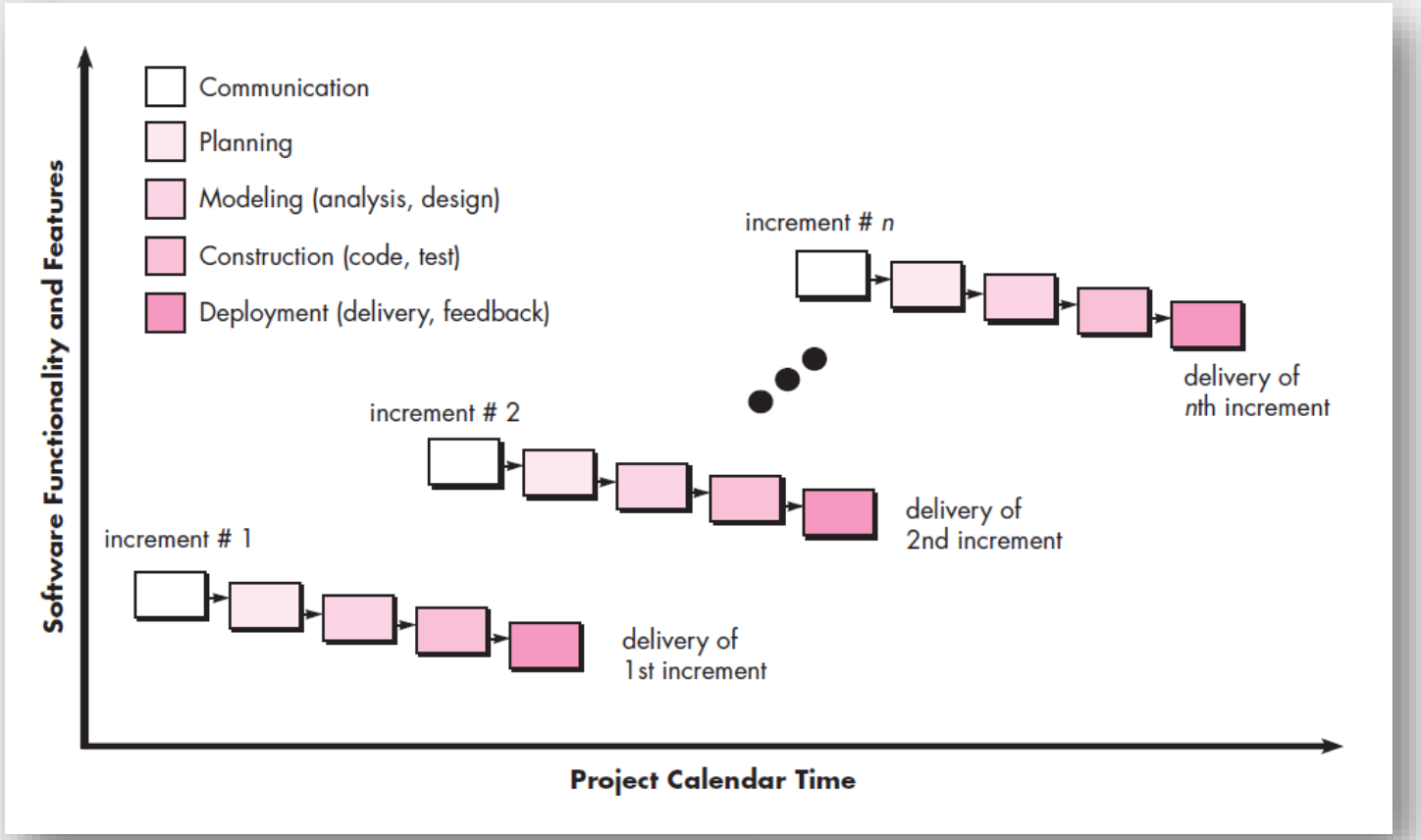
El modelo en cascada, a veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial para el desarrollo de software que comienza con la especificación de los requisitos del cliente y avanza a través de la planificación, el modelado, la construcción y la implementación, culminando en el soporte continuo del software completo. **Usualmente la calidad se integra como un hito en la fase de pruebas**.



Pressman (2010). Software Engineering 7th Ed. Pp. 39-40.

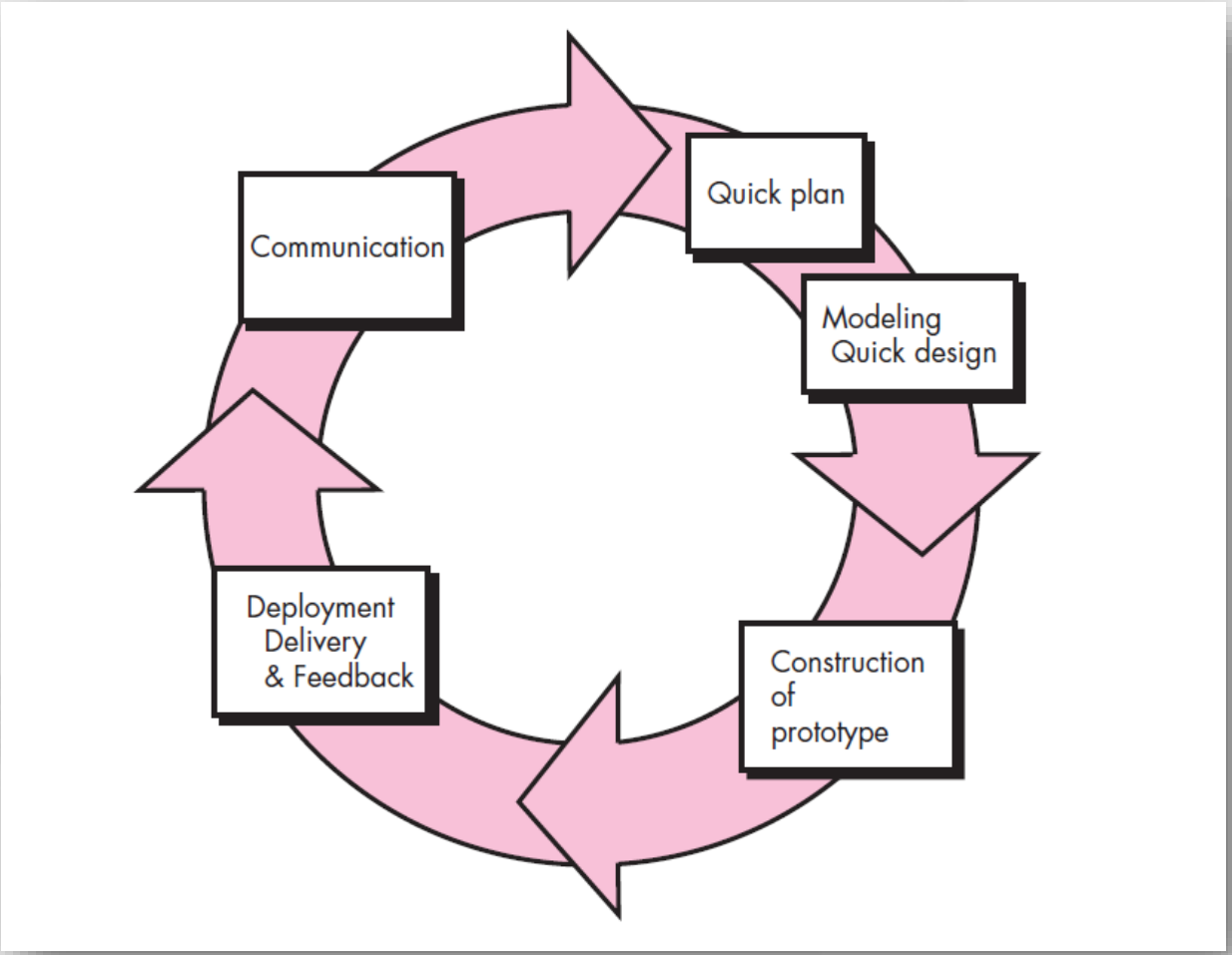
Modelo Incremental

Hay muchas situaciones en las que los requisitos iniciales del software están razonablemente bien definidos, pero el alcance general del esfuerzo de desarrollo impide un proceso puramente lineal. Puede haber una necesidad apremiante de proporcionar un conjunto limitado de funcionalidades de software a los usuarios rápidamente y luego perfeccionar y ampliar esa funcionalidad en versiones de software posteriores. En tales casos, puede elegir un modelo de proceso diseñado para producir el software en incrementos. La calidad se se debe asegurar con la entrega de cada incremento.



Pressman (2010). Software Engineering 7th Ed. Pp. 41-42.

Paradigma Prototipado



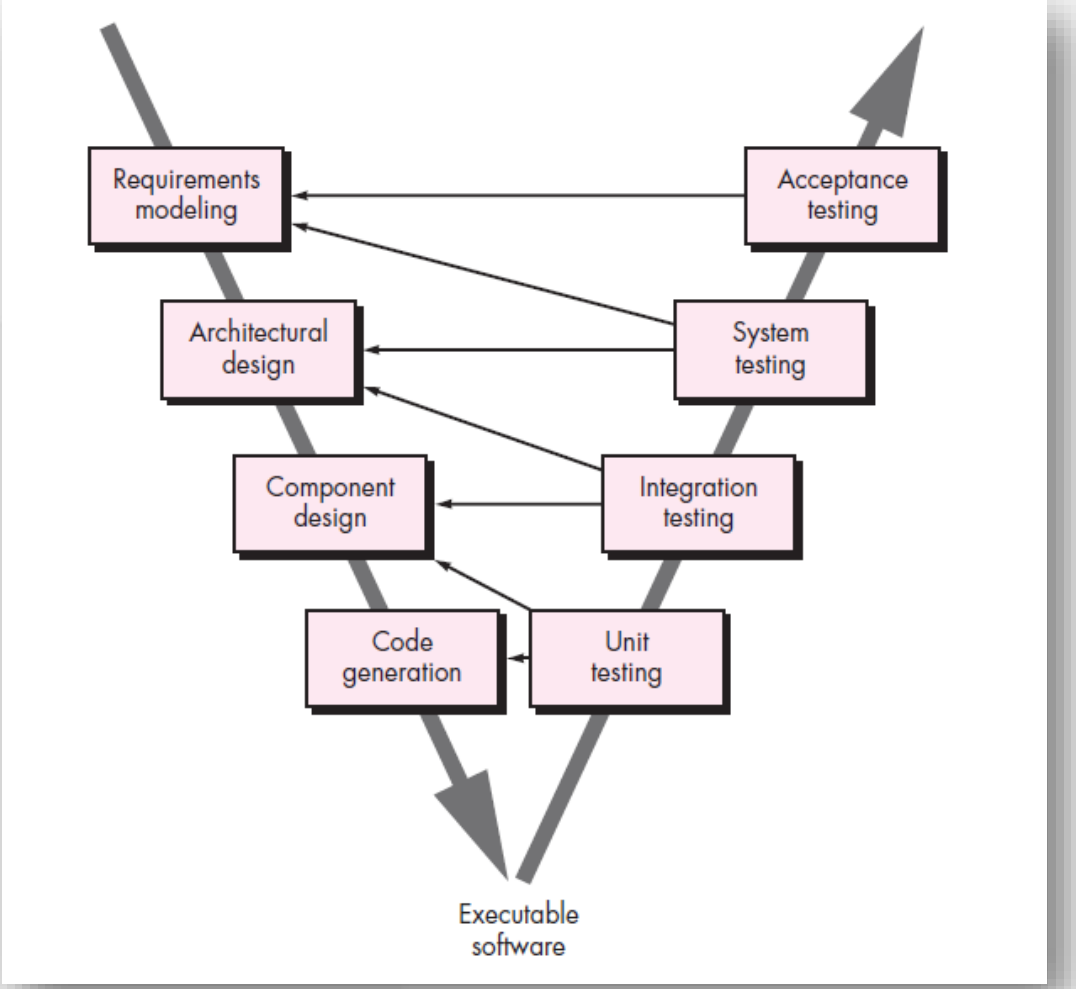
El prototipado por definición es una técnica que permite realizar y materializar diversas ideas de soluciones propuestas en un proyecto de diseño o rediseño de productos y servicios. El prototipado puede estar vinculado al recorrido completo de un servicio o bien a un punto de contacto específico.

En la ingeniería de software, permite validar de forma temprana los requerimientos con diversas entregas prototipadas, ya sea no funcionales o semi-funcionales. Esto es, desde dibujos, bocetos, hasta maquetas con navegación de fornt-end (Ej: Figma). **En estos casos los planes de calidad deben ir evolucionando con cada prototipo.**

Pressman (2010). Software Engineering 7th Ed. Pp. 43-44.

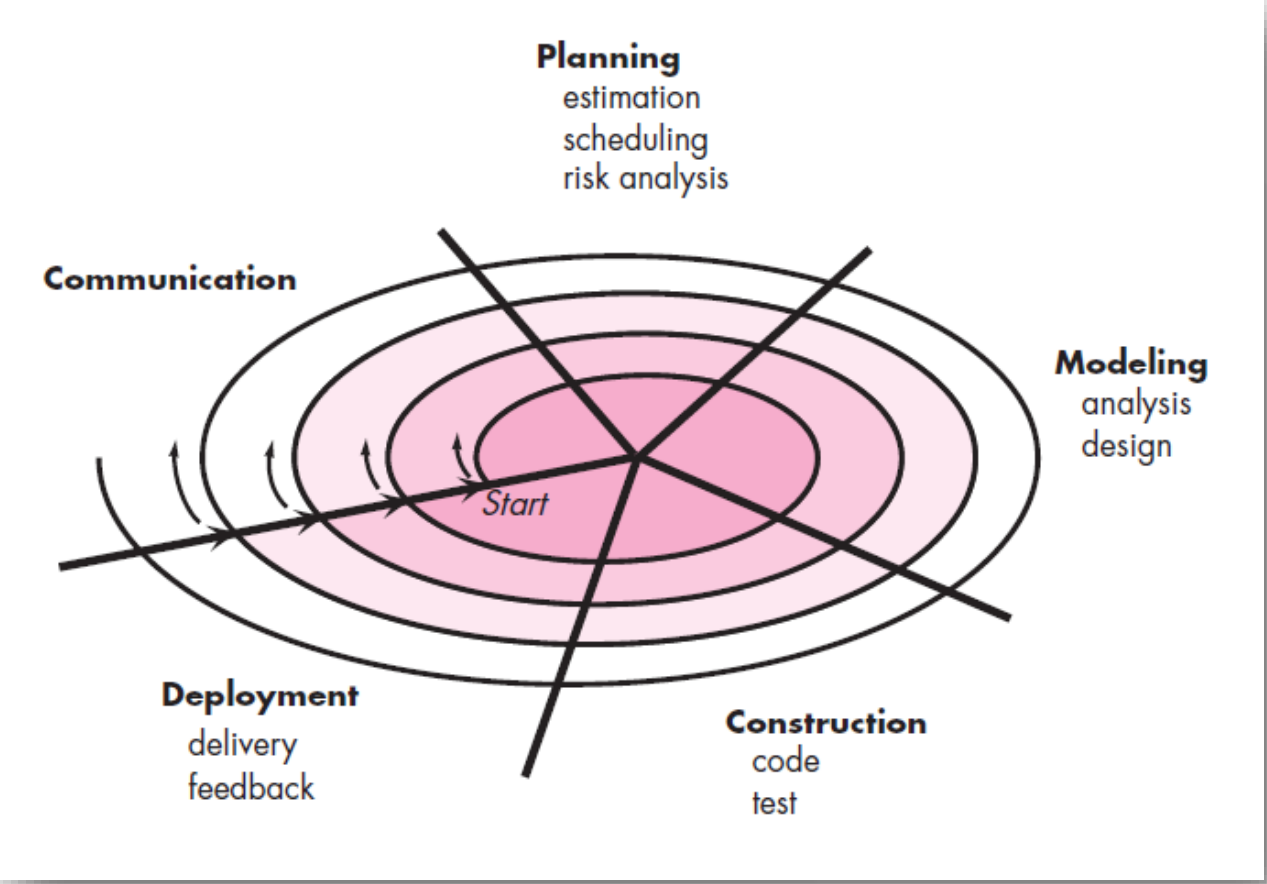
Modelo en “V”

Una variación en la representación del modelo en cascada se denomina modelo V. Describe la relación de las acciones de aseguramiento de la calidad con las acciones asociadas con la comunicación, el modelado y las actividades iniciales de construcción. A medida que un equipo de software avanza por el lado izquierdo de la V, los requisitos básicos del problema se refinan en representaciones progresivamente más detalladas y técnicas del problema y su solución. Una vez que se ha generado el código, el equipo sube por el lado derecho de la V, esencialmente realizando una serie de pruebas (acciones de control de calidad) que validan cada uno de los modelos creados a medida que el equipo baja por el lado izquierdo. En realidad, no existe una diferencia fundamental entre el ciclo de vida clásico y el modelo V. **El modelo V proporciona una forma de visualizar cómo se aplican las acciones de calidad, verificación y validación a trabajos de ingeniería anteriores.**



Pressman (2010). Software Engineering 7th Ed. Pp. 39-40.

Modelo Espiral

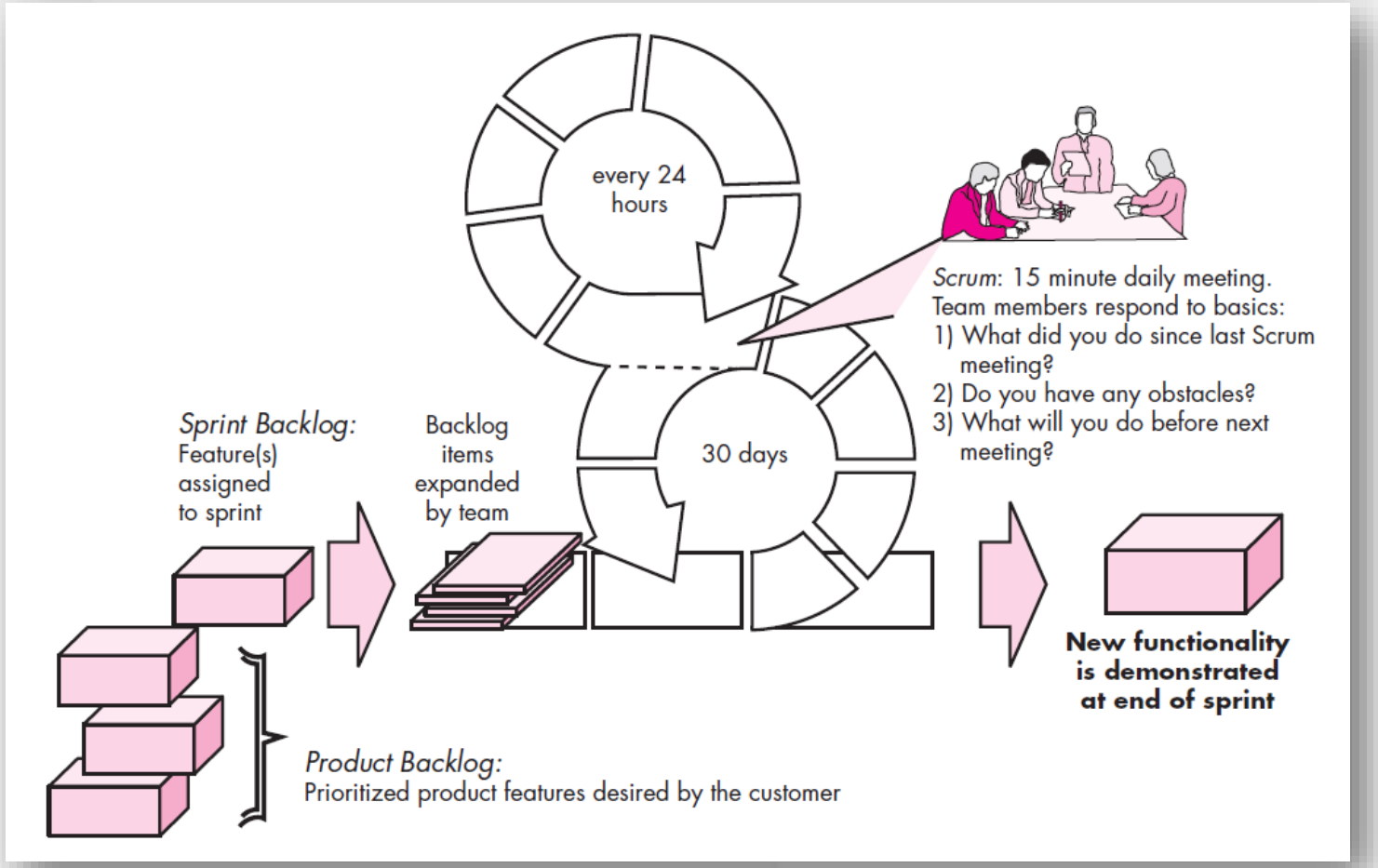


Un modelo en espiral se divide en un conjunto de actividades marco definidas por el equipo de ingeniería de software. Cada una de las actividades marco representa un segmento del camino en espiral. A medida que comienza este proceso evolutivo, el equipo de software realiza actividades que están implicadas en un circuito alrededor de la espiral en el sentido de las agujas del reloj, comenzando en el centro. El riesgo se considera a medida que se realiza cada evolución. Para cada paso evolutivo se anotan los hitos del punto de anclaje, una combinación de productos de trabajo y condiciones que se logran a lo largo del camino de la espiral. **El plan de calidad va evolucionando en cada iteración del espiral.**

Pressman (2010). Software Engineering 7th Ed. Pp. 43-44.

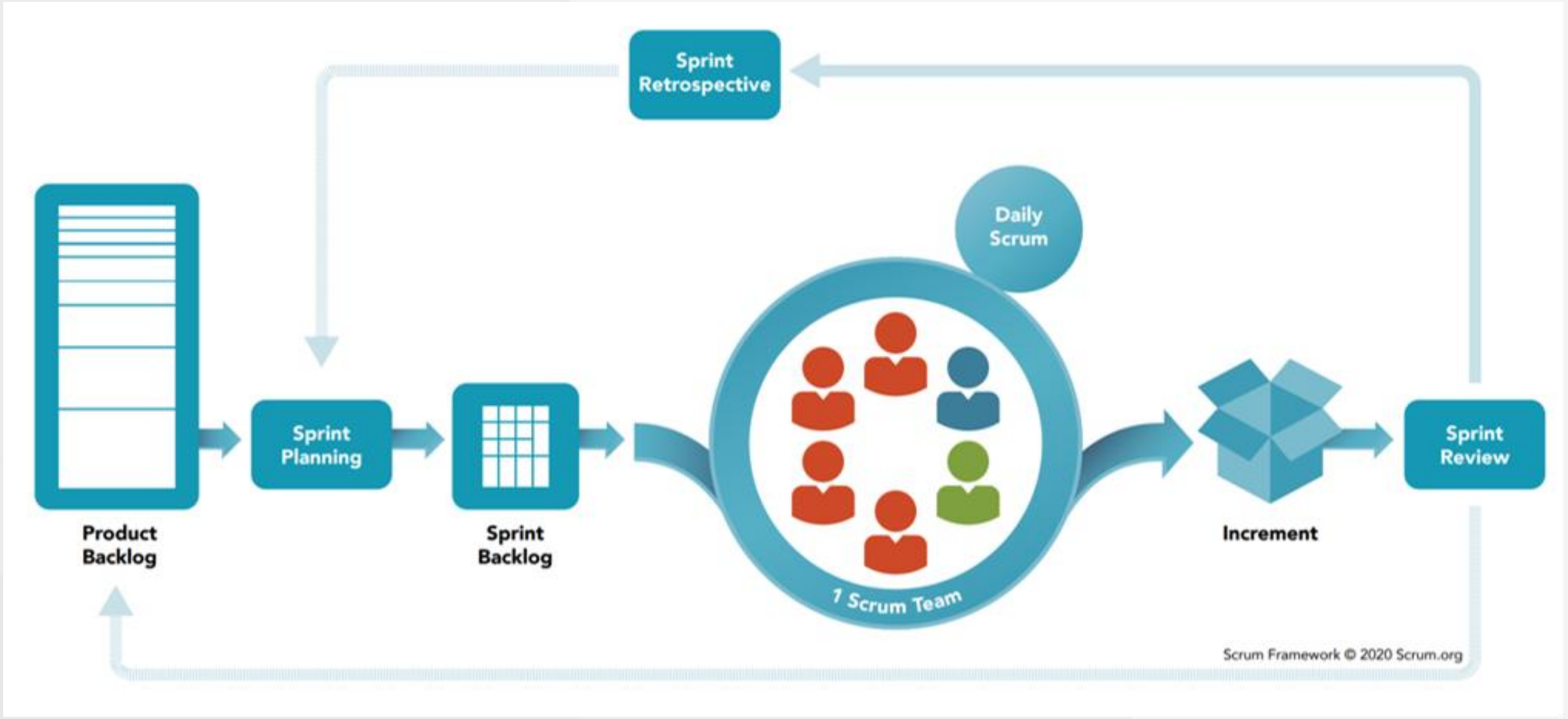
Marco de trabajo Scrum

Los principios de Scrum son consistentes con el manifiesto ágil (2001) y se utilizan para guiar las actividades de desarrollo dentro de un proceso que incorpora las siguientes actividades marco: requisitos, análisis, diseño, evolución y entrega. Dentro de cada actividad marco, las tareas de trabajo ocurren dentro de un patrón de proceso (que se analiza en el siguiente párrafo) llamado sprint. El trabajo realizado dentro de un sprint (el número de sprints necesarios para cada actividad del marco variará según la complejidad y el tamaño del producto) se adapta al problema en cuestión y el equipo Scrum lo define y, a menudo, lo modifica en tiempo real. **En este caso, el aseguramiento de la calidad se planifica mediante actividades específicas en el backlog.**



Pressman (2010). Software Engineering 7th Ed. Pp. 82-83.

¿Cómo funciona Scrum?



Estándares de calidad:
Buenas prácticas para la implementación de control de calidad.

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

ISO International Organization for Standardization

IEC INTERNATIONAL ELECTROTECHNICAL COMMISSION

IEEE
Advancing Technology for Humanity

Estándares de calidad

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Una vez adaptados al modelo de desarrollo: ¿Qué normas y estándares encontramos para la implementación del control de calidad?

Faro Medina, I. (2022). Los estándares de calidad del software más importantes. Hiberus.

ISO 9001 Gestión de la Calidad

- Esta norma para la implementación de un método o Sistema de Gestión de la Calidad (SGC). Acredita ante cualquier parte interesada la capacidad de una organización de satisfacer los requisitos del cliente.

ISO 10005:2018 Sistemas de Gestión de la calidad. Directrices para los planes de la calidad

- Ofrece las directrices para gestionar un plan de calidad que incremente la confianza, el control y la oportunidad de mejora durante todo el ciclo de vida. Se enfoca en el desarrollo, revisión, aceptación, aplicación y revisión del plan para adaptarlo a procesos, productos, proyectos o contratos.

ISO 33000 Calidad de los procesos de desarrollo de software

- También llamada Process Improvement and Capability Determination (SPICE), aporta unas líneas de trabajo para la evaluación de procesos software. Proporciona una base que permite evaluar el punto en el que se encuentra una empresa.

ISO 12207 Modelos de Ciclos de Vida del Software

- Se trata de un estándar para los procesos de ciclo de vida del software, entendidos como un conjunto de actividades y tareas relacionadas. Hace el recorrido desde que surge la necesidad o nueva idea hasta la deprecación o retirada del software.

ISO IEC IEEE 12207 Procesos de ciclo de vida de software

- Recomienda un marco común para los procesos de Ciclo de Vida del Software, desde la necesidad hasta la retirada. El propósito es proporcionar requisitos uniformes mínimos aceptables para la preparación y el contenido de los planes de aseguramiento de la calidad del software.

IEEE 730 – 2002 Standard for Software Quality Assurance Plans

- Este estándar define lo que es el software de alta calidad y es una recomendación para elaborar un Plan de Aseguramiento de la calidad de software (SQAP). Es utilizado en las fases de desarrollo y mantenimiento del ciclo de vida del software.

ISO IEC 25000

- Esta norma sustituye a ISO 9126 e ISO/IEC 14598. También conocida como SQuaRE (System and Software Quality Requirements and Evaluation), se trata de una familia de normas para evaluar la calidad del producto software. ¡La profundizamos la clase anterior!

ISO IEC IEE 29119 Norma para la documentación de prueba de software. Standard for Software Test Documentation

- Sustituye a IEE 829 – 1998. Está enfocada a la relación de las pruebas con las metodologías de desarrollo y el ciclo de vida software.

ISO IEC 20246 Ingeniería de Software

- Esta norma sustituye a IEE 1028. Establece un marco genérico para revisiones de productos de trabajo. Cualquier artefacto producido por un proceso puede ser considerado un producto de trabajo.

Ejemplo de buenas prácticas relevantes

IEEE 730-2014

ISO/IEC/IEEE 29119

La principal compilación de buenas prácticas en la implementación del control de calidad, la encontramos en el estándar IEEE 730: Plan de Aseguramiento de la calidad de software. Así también, se debe incluir el estándar ISO/IEC/IEEE 29119: Ingeniería de software y sistemas - Pruebas de software.

El primer estándar define lo que es el software de alta calidad y es una recomendación para elaborar un Plan de Aseguramiento de la calidad de software (SQAP). Es utilizado en las fases de desarrollo y mantenimiento del ciclo de vida del software. Por otra parte, define vocabulario, procesos, documentación, técnicas y un modelo de evaluación de procesos para pruebas que se pueden usar dentro de cualquier ciclo de vida de desarrollo de software.

IEEE proporciona una estructura de documentación que asegura que el plan de aseguramiento de calidad y el plan de pruebas de software, tengan los puntos esenciales según la buena práctica.

Resumen Norma

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Propósito y Alcance

- Generar evidencia justificada y confianza en productos de software
- Aplicable a cualquier proyecto, sin importar tamaño o complejidad
- Armonizado con ISO/IEC/IEEE 12207:2008

Conceptos Clave del SQA

Calidad como Conformidad

Grado en que un producto cumple con requisitos establecidos

Gestión de Riesgos

Actividades proporcionales al riesgo del producto

Independencia SQA

Función independiente técnica, gerencial y financieramente

Aseguramiento del Producto

Evalúa que los productos de software (código, documentación, etc.) se ajusten a sus requisitos específicos.

Enfoque: Artefactos y Entregables

Aseguramiento del Proceso

Verifica que las actividades del proyecto se ejecuten de acuerdo con los planes y procesos definidos.

Enfoque: Metodologías y Procedimientos

Ejemplo de buenas prácticas relevantes

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

Implementación del Proceso SQA

6 Actividades (5.3)

Actividades Clave:

- Establecer procesos SQA
- Coordinar con procesos relacionados
- Documentar planificación SQA
- Ejecutar el Plan SQA
- Gestionar registros SQA
- Evaluar independencia organizacional

Aseguramiento del Producto

5 Actividades (5.4)

Evaluaciones:

- Conformidad de planes
- Conformidad del producto
- Aceptabilidad del producto
- Soporte del ciclo de vida
- Medición de productos

Aseguramiento del Proceso

5 Actividades (5.5)

Verificaciones:

- Conformidad de procesos del ciclo de vida
- Conformidad de entornos
- Procesos de subcontratistas
- Medición de procesos
- Habilidades del personal

Plan de Aseguramiento de la Calidad del Software (SQAP)

Estructura Normativa:

- Propósito y alcance
- Visión general del plan SQA
- Actividades, resultados y tareas
- Consideraciones adicionales

Aplicabilidad Específica:

Metodologías Ágiles
Integración en ciclos iterativos

Industrias Reguladas
Dispositivos médicos, energía nuclear

Estándar ISO/IEC/IEEE 29119

¡Estos son los puntos esenciales que debe tener todo plan y documentación para implementar un aseguramiento de calidad en un desarrollo de software, con su respectivo plan de pruebas!

- **ISO/IEC/IEEE 29119-1:2013, Parte 1**
  - Conceptos y definiciones
- **ISO/IEC/IEEE 29119-2:2013, Parte 2**
  - Procesos de prueba
- **ISO/IEC/IEEE 29119-3:2013, Parte 3**
  - Documentación de prueba
    - Documentación del proceso de prueba organizacional
    - Documentación del proceso de gestión de pruebas
    - Documentación del proceso de prueba dinámica
- **ISO/IEC/IEEE 29119-4:2015, Parte 4**
  - Técnicas de prueba
- **ISO/IEC/IEEE 29119-5:2016, Parte 5**
  - Pruebas basadas en palabras clave

ISO/IEC/IEEE 29119-1:2013. Standards catalogue. International Organization for Standardization.

Ejercicio práctico para aplicar lo aprendido

La siguiente plantilla representa un *template* para casos de prueba siguiendo orientaciones del estándar ISO/IEC/IEEE 29119. Defina un desarrollo de software ficticio, lo más breve y sencillo posible. Tome el ejemplo de la primera fila y proponga al menos 6 casos de prueba relevantes para la aplicación seleccionada.

Project Name	New Website
Module Name	Login
Reference Document	N/A
Created By	Joe Smith
Date of Creation	1/5/2024
Date of Review	1/10/2025

Test Case ID	Test Case Scenario	Test Case	Pre-Conditions	Test Steps	Test Data	Expected Results
_001	Make sure login is working	Enter valid user ID and valid PW	Need valid email account	1) Enter user ID 2) Enter PW 3) login	<valid user ID> <valid PW>	Successful login

Test Case Template (s.f.). ProjectManager.

Referencias Bibliográficas

- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001) Manifesto for Agile Software Development.
- Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. IEEE Computer, 21 (5), 61–72.
- Brown, S. (2012). Software Architecture for Developers. Pp. 34.
- Faro Medina, I. (2022). Los estándares de calidad del software más importantes. Hiberus. https://www.hiberus.com/crecemos-contigo/los-estandares-de-calidad-del-software-mas-importantes/
- Pressman, R. (2010). Software Engineering 7th Ed. Chapter 2.
- Sommerville, I. (2011). Software Engineering 9th Ed. Figure 2.11. Pp. 48-49.
- ISO/IEC/IEEE 29119-1:2013. Standards catalogue. International Organization for Standardization.
- IEEE 730-2014. Standard for Software Quality Assurance Processes.
- Test Case Template (s.f.). ProjectManager. https://www.projectmanager.com/templates/test-case-template

UNIVERSIDAD SAN SEBASTIAN
VOCACIÓN POR LA EXCELENCIA

¡Tus consultas son importantes!

Espacio para preguntas