Conceptos de Aseguramiento de la Calidad en Software

Calidad de un producto de Software – ISO 25000

La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas.

Aseguramiento de la Calidad en Software

A set of activities that define and assess the adequacy of software processes to provide evidence that establishes confidence that the software processes are appropriate for and produce software products of suitable quality for their intended purposes. A key attribute of SQA is the objectivity of the SQA function with respect to the project. The SQA function may also be organizationally independent of the project; that is, free from technical, managerial, and financial pressures from the project.

IEEE 730

Aseguramiento de la Calidad vs Testing

Aseguramiento de la Calidad

Enfoque PREVENTIVO

Procesos de Mejora

“Si un buen proceso se sigue correctamente, generará buenos productos”

“Parte de la gestión de la calidad orientada a proporcionar ***confianza*** en que se cumplirán los requisitos de la calidad.” ISO 9000:2005

Es responsabilidad de TODOS en el proyecto

Testing

Forma de Control de Calidad

Enfoque CORRECTIVO, orientado al producto, para permitir alcanzar los niveles adecuados de calidad

UNIVERSIDAD ORT Uruguay

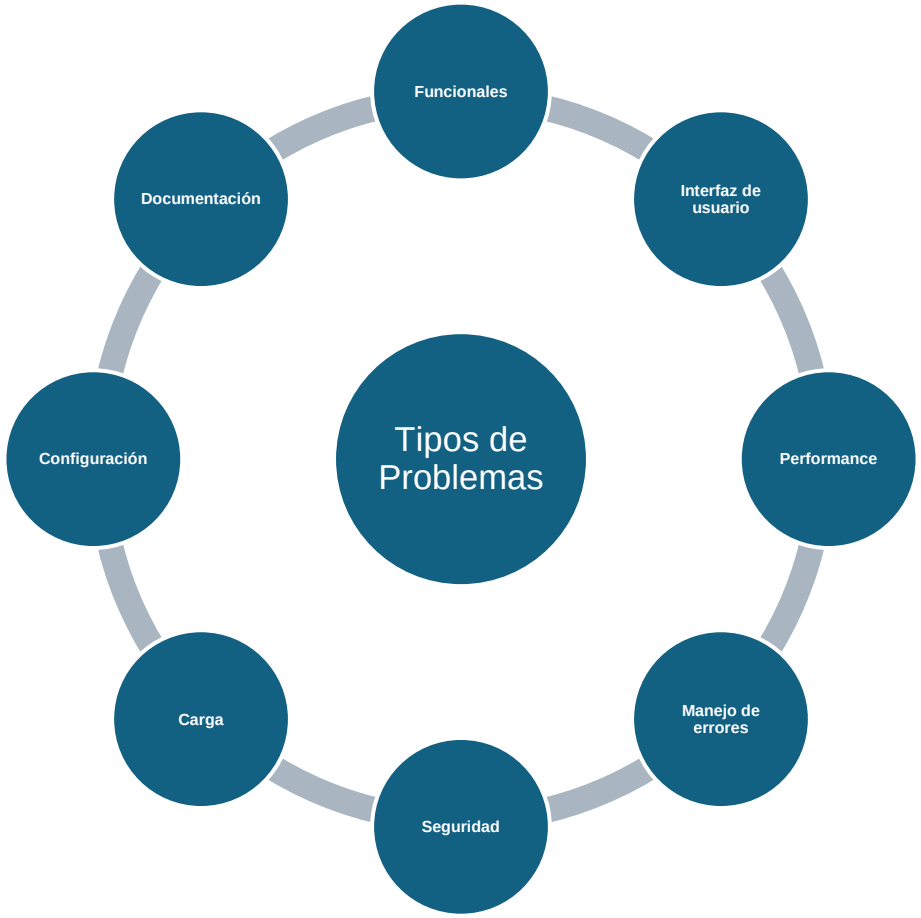
Facultad de Ingeniería

Gestión de Calidad del Software

- Aseguramiento de la Calidad: definición de un marco de trabajo, procedimientos y estándares organizacionales que conducen a producir software de alta calidad.
- Planificación de la Calidad: selección de procedimientos y estándares adecuados dentro del marco de Aseguramiento de la Calidad para un proyecto de software.
- Control de Calidad: definición y aplicación de procesos que garanticen los procedimientos y estándares aplicados por el equipo de software.

¿Por qué Aseguramiento de la Calidad?

- Para poder encontrar, reportar y corregir problemas en el software.
- ¿Qué se entiende por problema?
  - Discrepancia entre la especificación de requerimientos y el comportamiento de programa (se parte de la base de que la especificación es correcta)



“Quality is not only an important concept, it is also free. And it is not only free, it is the most profitable product that we produce! The real question is not how much a quality management system costs, but what is the cost of not having one.”

Harold Geneen
CEO ITT Corporation

Validación y Verificación

Validación

¿Estamos construyendo el producto correcto?

- Actividades para determinar si las necesidades del cliente están siendo cubiertas con los requerimientos especificados

Verificación

¿Estamos construyendo correctamente el producto?

- Actividades para determinar que el software cumple con las especificaciones en cada etapa del proceso

Tipos de Actividades del Aseguramiento de la Calidad

Actividades preventivas.

Planificación de la calidad
Análisis causal.
Revisiones y auditorías.

Actividades correctivas.

Verificación y validación (Prueba de productos).

Actividades registrales.

Registro de resultados y redacción de informes
Recolección de métricas: de productividad, de re trabajo, de fallas.

Costos de las Actividades de Aseguramiento de la Calidad

- Costo de prevención
  - Planificación, organización y capacitación
- Costo de la evaluación
  - Revisiones, auditorías, verificación y validación.
- Costo de fallas
  - Internas
    - Reparación
  - Externas
    - Resolución de quejas, devolución y sustitución de productos, soporte, trabajo de garantía

Costos Ocultos de la NO calidad

Retrabajos
Productos de Segunda
Gatos en garantías
Proyectos que no terminan
Clientes perdidos
Horas extras para corregir errores
Desmotivación
Tiempo en apagar incendios
Pérdida de imagen
Falta de participación


ORT
UNIVERSIDAD ORT
Uruguay
Facultad de
Ingeniería
Pruebas

Errores, Defectos, Fallas y Causas Raíz

- Las personas comenten errores por distintos motivos (cansancio, desconocimiento, complejidad de los elementos con que trabajan, etc), que introducen defectos en los productos, que se manifiestan en fallas.
- Causas Raíz son las razones fundamentales de la ocurrencia del problema, por qué se originó.

Objetivos de la Prueba

- Evaluación de entregables
- Detección de defectos
- Aseguramiento de cobertura de pruebas
- Reducción de riesgos de calidad inadecuada del software
- Verificación de cumplimientos de objetivos
- Verificación de cumplimiento de normativas o acuerdos contractuales, legales o regulatorios
- Aportar confianza en el producto

Pruebas y Debugging

Testing

- Busca manifestar fallas causadas por defectos
- Detecta defectos (mal funcionamientos) en los productos
- Medio efectivo en costo para identificar problemas

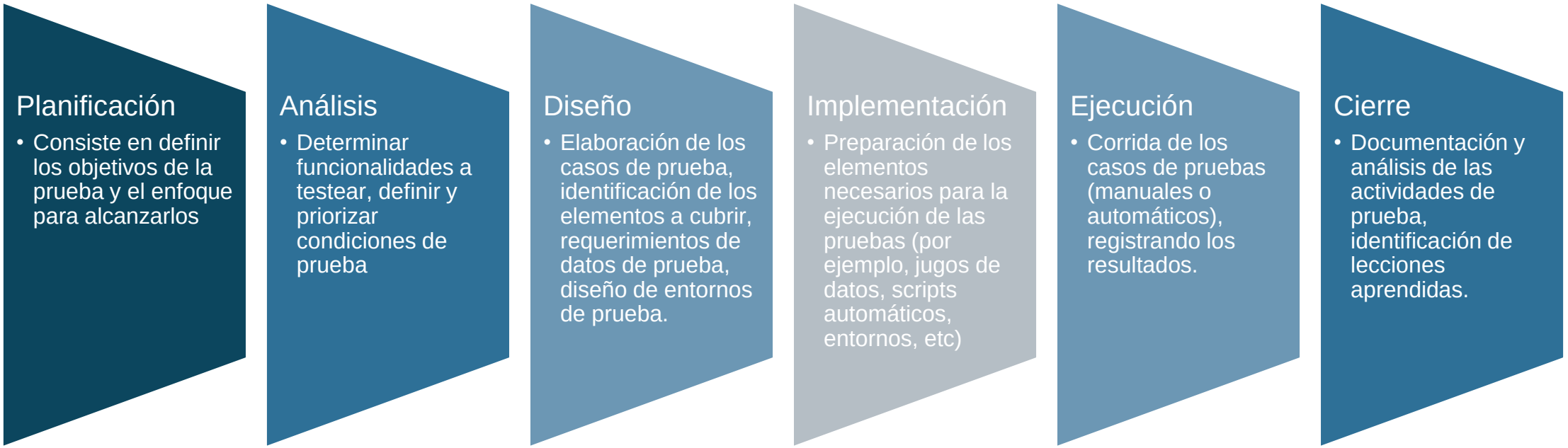
Debuggin

- Busca encontrar las causas de las fallas
- Reproduciendo las fallas
- Diagnosticando
- Corrigiendo la causa

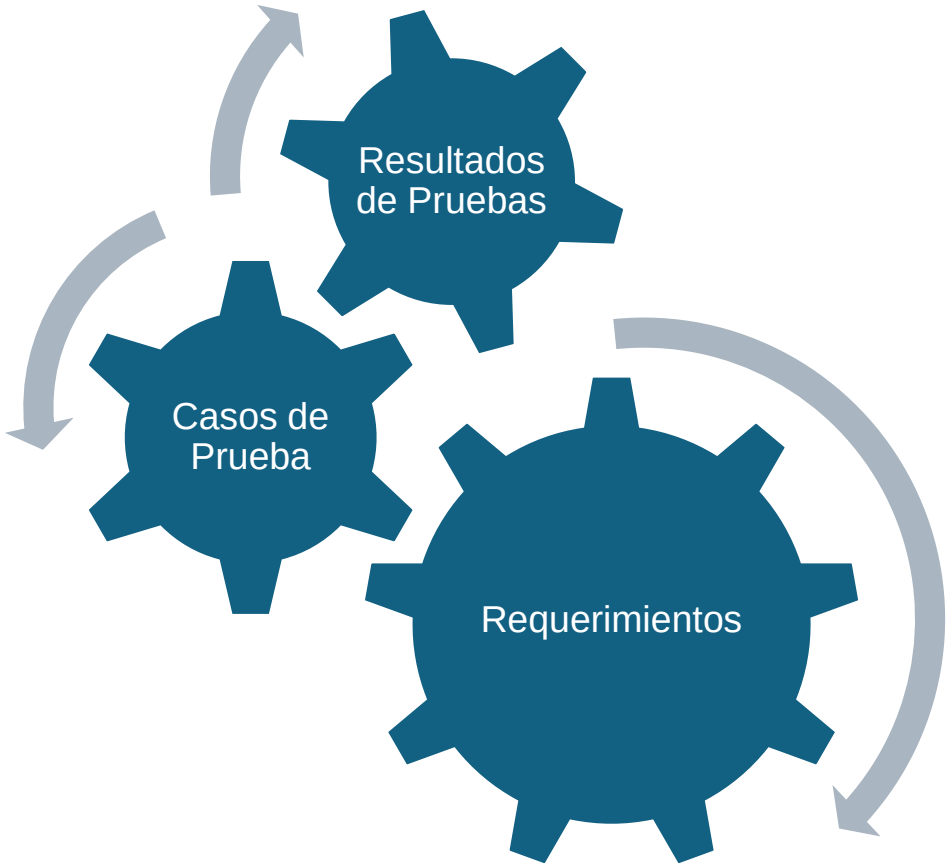
Principios del Testing

- El testing demuestra la presencia, no la ausencia de defectos.
- El testing exhaustivo es imposible
- El testing temprano ahorra tiempo y dinero
- Los defectos tienden a agruparse en “clusters”
- Las pruebas “pierden eficacia”
- La prueba depende del contexto
- Falacia de la Ausencia de Defectos

Actividades del Testing



Trazabilidad



- Casos de Prueba <> Requerimientos para garantizar COBERTURA
- Resultados de Prueba <> Casos de Prueba para evaluar riesgos residuales en el elemento testeado

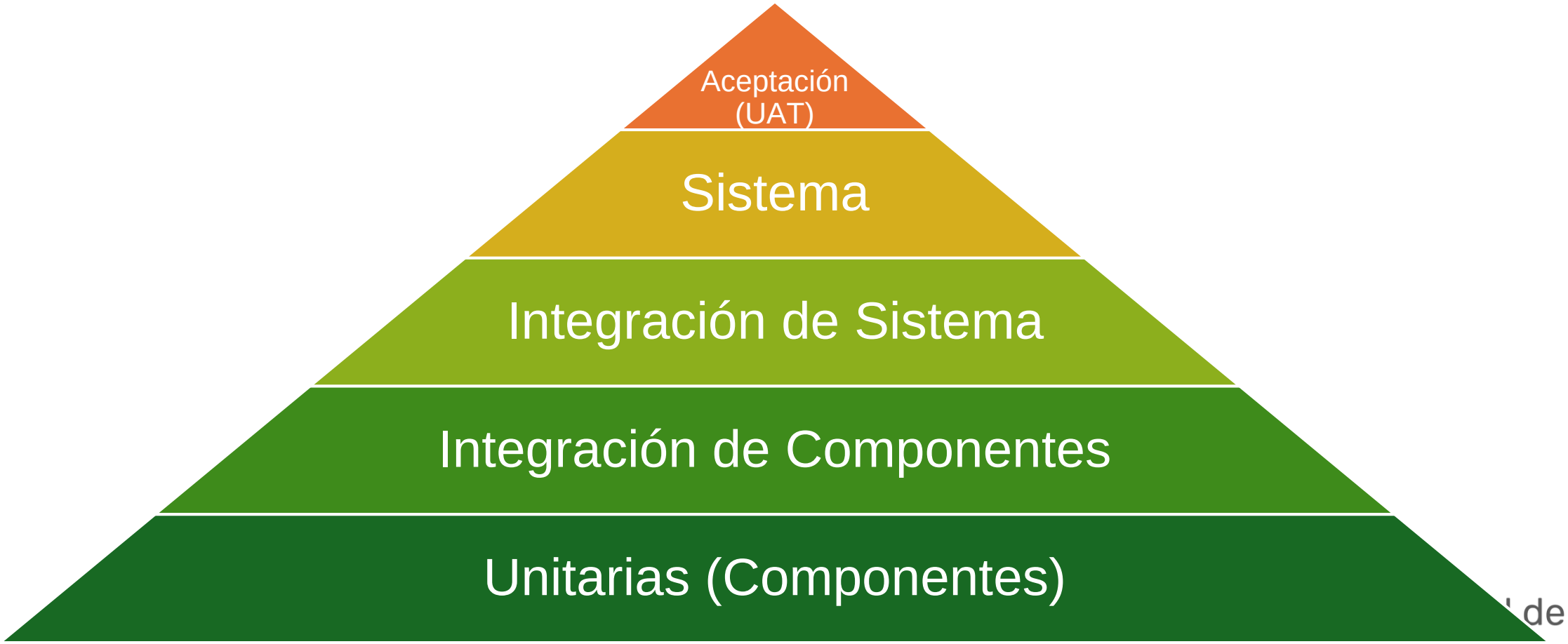
Factores que inciden en el proceso de Prueba

- Interesados: expectativas, necesidades, participación, etc.
- Dominio de negocio: criticidad, riesgos, necesidades de mercado, legislación específica, etc.
- Factores técnicos: tipo de software, arquitectura, tecnología, etc.
- Restricciones de proyecto: tiempos, presupuesto, recursos, etc.
- Modelo de ciclo de vida (predictivo, iterativo, etc.)
- Herramientas: disponibilidad, usabilidad, etc.

Buenas Prácticas para el Testing

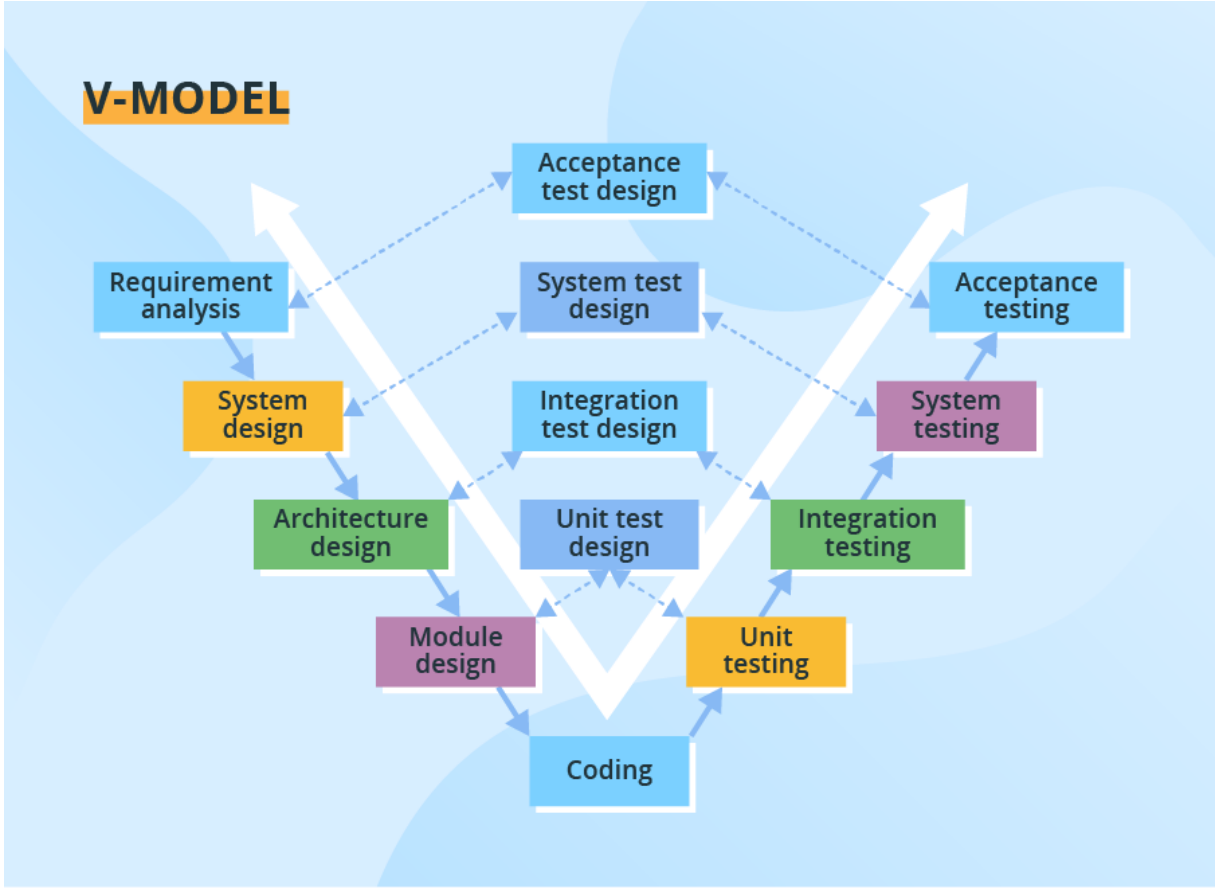
- Enfoque de UN EQUIPO: todos participan efectivamente para lograr los objetivos del proyecto.
- Independencia del Testing
- Para cada actividad de desarrollo de software, debe haber una actividad correspondiente de prueba del mismo.
- Los diferentes niveles de prueba tendrán objetivos específicos, permitiendo mayor cobertura y evitando redundancia.
- El testing deberá comenzar en fases tempranas (shift-left), a la par de las fases de desarrollo.

Niveles de Prueba



Modelo en V

- Cada etapa en el proceso de desarrollo debe tener su correspondiente nivel de pruebas.



Pruebas Unitarias

- Son pruebas de software que validan el funcionamiento correcto de una unidad individual de código (generalmente una función o un método) de manera aislada.
- Objetivos:
  - Verificar Funcionalidad: Asegurar que cada unidad de código funciona según lo esperado.
  - Detectar Errores Tempranos: Identificar y corregir errores en las primeras etapas del desarrollo.
  - Facilitar el Mantenimiento: Simplificar la detección de errores en futuras modificaciones del código.
  - Documentación del Código: Proveer ejemplos concretos de cómo se debe utilizar una unidad de código.

Pruebas de Integración de Componentes

- Son pruebas de software que se enfocan en verificar la interacción y la integración correcta entre múltiples componentes o módulos del sistema, corroborando la integración y la comunicación entre unidades de código específicas, identificando y corrigiendo problemas en las interfaces y la interacción entre ellas.
- Objetivos:
  - Verificar Interacciones: Asegurar que los componentes funcionan correctamente juntos.
  - Detectar Problemas de Integración: Identificar y corregir problemas en las interfaces y la comunicación entre componentes.
  - Validar Flujos de Datos: Comprobar que los datos se transmiten y procesan correctamente entre componentes.
  - Garantizar la Cohesión del Sistema: Asegurar que el sistema como un todo funciona de manera coherente y eficiente.

Pruebas de Integración de Sistema

- Son pruebas que se enfocan en verificar la interacción y la integración correcta entre los distintos componentes y subsistemas del sistema completo, corroborando la coherencia y la fiabilidad del sistema integrado.
- Objetivos:
  - Validar Interacciones Complejas: Comprobar que los componentes y subsistemas interactúan correctamente.
  - Identificar Problemas de Interfaz: Detectar errores en las interfaces entre componentes y subsistemas.
  - Evaluar el Funcionamiento del Sistema Completo: Verificar que el sistema completo cumple con los requisitos y expectativas del cliente.
  - Garantizar la Coherencia y la Fiabilidad: Asegurar que el sistema integrado funciona de manera coherente y fiable en todas las condiciones.

Pruebas de Sistema

- Son pruebas de software que se centran en validar todo el sistema como una entidad completa, verificando que cumpla con los requisitos y expectativas del cliente.
- Objetivos:
  - Validar Cumplimiento de Requerimientos: Confirmar que el sistema cumple con todos los requisitos funcionales y no funcionales especificados.
  - Evaluar la Usabilidad: Evaluar la facilidad de uso y la experiencia del usuario del sistema.
  - Verificar la Integración: Comprobar que todas las partes del sistema funcionan juntas correctamente.
  - Asegurar la Calidad Global: Garantizar que el sistema cumple con los estándares de calidad y rendimiento esperados.

Pruebas de Aceptación de Usuario (UAT)

- Son pruebas realizadas por usuarios finales para evaluar si el sistema cumple con sus necesidades y expectativas antes de su implementación en producción.
- Objetivos:
  - Validación del Sistema: Confirmar que el sistema satisface los requisitos del usuario y cumple con su propósito.
  - Verificación de la Usabilidad: Evaluar la facilidad de uso y la experiencia del usuario al interactuar con el sistema.
  - Identificación de Problemas Finales: Detectar errores o deficiencias que pueden haber pasado desapercibidos en pruebas anteriores.

Tipos de Prueba

Funcionales

Evalúa la funcionalidad que cada parte del sistema debe brindar.

Objetivo: comprobar corrección, completitud, propiedad de las funcionalidades

No Funcionales

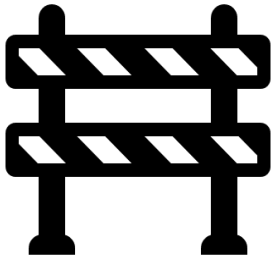
Evalúan características no funcionales

- Performance
- Compatibilidad
- Usabilidad
- Confiabilidad,
- Seguridad
- Portabilidad
- Mantenibilidad

Pruebas de Regresión y Confirmación

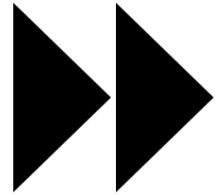
- Regresión: para confirmar que no hubo impactos negativos por los cambios introducidos en esta versión
- Confirmación: para confirmar que un defecto ha sido exitosamente resuelto

Pruebas Estáticas y Dinámicas



Pruebas Estáticas

Definición: Pruebas que se realizan sin ejecutar el código.



Pruebas Dinámicas

Definición: Pruebas que se realizan ejecutando el código.

Pruebas Estáticas

- Se refiere a las instancias de control en las que NO se ejecuta el software, sino que se revisan (manual – Revisiones - o automáticamente – Análisis Estático) los códigos, especificaciones de procesos o arquitectura, etc.
- Permite identificar problemas antes de la ejecución dinámica de los casos de prueba y no requiere juegos de datos.
- Implica, por ejemplo, controlar la estructura interna de los productos, cumplimiento de estándares de documentación, etc.
- Se puede aplicar a entregables no-ejecutables (documentación, diagramas, especificaciones, por ejemplo)
- Permite evaluar características que no dependen del código ejecutable (por ejemplo, mantenibilidad)

Ejemplos de Defectos – Testing Estático

- Defectos en requerimientos (inconsistencias, ambigüedades, contradicciones, omisiones, inexactitudes, duplicados)
- Defectos de diseño (estructuras de base de datos ineficientes, modularización pobre)
- Defectos de codificación (variables sin definir, sin inicializar, código inalcanzable o duplicado)
- Desviaciones de estándares (nomenclatura, codificación)
- Especificaciones incorrectas de interfaz (parámetros, problemas de definición de funciones)

Pruebas Estáticas: Actividades del Proceso de Revisión

Tipos de Revisiones

Informal	Walk-through	Revisión Técnica	Inspección
• No siguen un proceso definido ni requiere un documento formal	• Implica una recorrida, hecha por el autor de un producto, sobre el mismo, para el análisis por los revisores	• Realizada por revisores calificados y liderada por un moderador con el fin de lograr consenso y tomar decisiones sobre un problema técnico, identificando anomalías y evaluando calidad.	• Instancia formal de revisión, siguiendo el proceso completo de las revisiones, con el objetivo de identificar la mayor cantidad posible de anomalías

Testing Dinámico

- Se refiere a las instancias de control que implican la ejecución misma del producto, ejercitando casos de pruebas, para verificar el funcionamiento.

Técnicas de Prueba

Caja Negra – basadas en el análisis del comportamiento del objeto (resultados vs entradas)

Particionamiento Equivalente: divide los juegos de datos en particiones equivalentes, basados en el concepto de que todos los elementos de la misma partición deberían tener el mismo comportamiento.

Análisis de Bordes: parte de la base de que en general es más factible que haya defectos en los casos de borde de las particiones, más que en los valores normales.

Tablas de Decisión: se utilizan para evaluar, por ejemplo, reglas de negocios con lógicas complejas, donde la combinación de diferentes valores se traduce en distintos resultados.

Transiciones de Estado: analiza cómo se comporta el sistema al modelar los cambios de estado del mismo.

Caja Blanca – basadas en el análisis de la estructura interna del objeto

Testing de Sentencias – cubre los comandos ejecutables del código

Testing de Bifurcaciones – analiza los caminos que se siguen en las condiciones del código

Experimental – basadas en el conocimiento y experiencia de los testers

Error Guessing: busca anticipar la ocurrencia de fallas, a partir de la experiencia

Testing Exploratorio: ls casos de prueba se diseñan, ejecutan y evalúan mientras el tester está aprendiendo sobre el objeto

Testing basado en listas de verificación

Casos de Prueba

- Un caso de prueba es un conjunto específico de condiciones o variables bajo las cuales un tester determinará si un sistema o una de sus características está funcionando correctamente.

- Elementos clave:

  - Descripción clara de lo que se está probando.
  - Pasos específicos para llevar a cabo la prueba.
  - Resultados esperados.

¿Qué Debería Tener un Caso de Prueba?

- ID del Caso de Prueba: Un identificador único.
- Título: Nombre breve y descriptivo del caso de prueba.
- Descripción: Explicación detallada del propósito de la prueba.
- Identificador del Requerimiento referido: Trazabilidad al requerimiento que se evalúa.
- Precondiciones: Condiciones que deben cumplirse antes de ejecutar la prueba.
- Pasos para Ejecutar: Lista detallada de acciones para realizar la prueba.
- Datos de Prueba: Información específica necesaria para la prueba.
- Resultados Esperados: Descripción de lo que debería ocurrir si la prueba es exitosa.
- Resultados Actuales: Lo que realmente ocurre cuando se ejecuta la prueba.
- Estado: Resultado final de la prueba (Ej. Pasó, Falló, Bloqueado).

¿Cómo Debería Ser un Caso de Prueba?

- Claro y Conciso: La descripción y los pasos deben ser fáciles de entender.
- Específico: Debe centrarse en un objetivo o funcionalidad específica.
- Reproducible: Cualquier persona debería poder seguir los pasos y obtener los mismos resultados.
- Independiente: Debería poder ejecutarse de forma aislada sin depender de otros casos de prueba.
- Objetivo: Los resultados esperados deben ser medibles y no subjetivos.

Resultados Positivos y Negativos

- Resultados Positivos: Verifican que la funcionalidad se comporta como se espera cuando se cumplen las condiciones correctas.
- Ejemplo: El sistema redirige al usuario a la página principal tras iniciar sesión con credenciales válidas.
- Resultados Negativos: Verifican cómo el sistema maneja condiciones incorrectas o excepcionales.
- Ejemplo: El sistema muestra un mensaje de error si se ingresan credenciales inválidas.
- Beneficios:
  - Cobertura Completa: Asegura que todas las posibles condiciones son probadas.
  - Identificación de Defectos: Ayuda a descubrir errores y casos no contemplados.
  - Mejor Experiencia de Usuario: Garantiza que el sistema maneja errores de manera adecuada y proporciona retroalimentación útil al usuario.

Ejemplo de Caso de Prueba

ID	TC001
Titulo	Verificación de Inicio de Sesión
Descripción	Verificar que un usuario puede iniciar sesión con credenciales válidas.
Precondiciones	El usuario debe tener una cuenta registrada.
Pasos para Ejecutar	
Navegar a la página de inicio de sesión.	
Ingresar nombre de usuario y contraseña válidos.	
Hacer clic en el botón "Iniciar sesión".	
Datos de Prueba	
Usuario	user@example.com
Contraseña	Password123
Resultados Esperados	El usuario es redirigido a la página principal de su cuenta.
Positivo	El sistema redirige al usuario a la página principal.
Negativo	El sistema muestra un mensaje de error si las credenciales son inválidas.
Resultados Actuales	(Se llenará tras ejecutar la prueba)
Estado	(Se llenará tras ejecutar la prueba)


UNIVERSIDAD ORT
Uruguay
Facultad de
Ingeniería
Planeando la Calidad

Plan de Aseguramiento de la Calidad

- Cada proyecto de desarrollo y mantenimiento debe tener un plan de SQA especificando sus metas, tareas de SQA a desarrollar, los estándares a utilizar, los procedimientos y estructura organizacional
- Se debe considerar el tipo de proyecto y el grupo que lo desarrolla para seleccionar las actividades de aseguramiento de la calidad.
- Los estándares que se aplican deben ser concordante con los atributos de calidad del sistema.

Planificación de Pruebas

- Plan de Pruebas: describe objetivos, recursos y procesos de un proyecto de pruebas

Contexto: alcance, objetivos, limitaciones

Supuestos y restricciones

Interesados: roles, responsabilidades, necesidades de entrenamiento

Comunicación: modos y frecuencias de comunicaciones, plantillas de documentación y reportes

Registro de riesgos

Enfoque de pruebas: niveles, tipos, técnicas de pruebas, entregables de testing, criterios de entrada y salida de pruebas, métricas, requerimientos de entorno

Cronograma y presupuesto

Contenido del plan de SQA (IEEE)

- Propósito y referencias
- Gerenciamiento (Tareas y responsabilidades)
- Documentación
- Estándares, prácticas y métricas
- Revisiones y auditorías
- Pruebas
- Gestión del cambio (SCM)
- Reporte de problemas y acciones correctivas
- Herramientas, técnicas y metodologías
- Recolección, mantenimiento y retención de registros

Plan de Pruebas

- Define qué, cómo, dónde se va a probar y quién lo va a hacer
- Tener en cuenta:
  - Requerimientos funcionales y no funcionales, ¿es un sistema nuevo? ¿modifica uno existente?
  - Alcance de las Pruebas:¿Qué funcionalidades hay que probar? (nuevas y ya existentes) ¿Cuáles NO hay que probar?
  - Estrategia de Pruebas
  - Criterios de inicio, finalización y suspensión de pruebas
  - Ambientes/Entornos requeridos
  - Conocimientos necesarios para realizar las pruebas (entrenamiento)
  - Cronograma de pruebas (con responsables)


ORT
UNIVERSIDAD ORT
Uruguay
Facultad de
Ingeniería
Midiendo la Calidad

Mediciones y Métricas

Para cada atributo elegido del proceso o del producto se determina un valor a alcanzar y luego se lo compara con lo obtenido

Medida: proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos de los atributos de un proceso o producto. Son los SENSORES que detectan “problemas”.

Medición: es el acto de determinar una medida. Permite mejorar el proceso de software ayudando en la planificación, seguimiento y control de un proyecto y permite evaluar la calidad del producto que se produce.

Métrica: según IEEE es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

UNIVERSIDAD ORT Uruguay | Facultad de Ingeniería

Métricas de Testing

- Algunas métricas comunes incluyen:
  - Grado de Avance de proyecto: tareas terminadas, uso de recursos, esfuerzo en testing
  - Avance de Pruebas: casos de prueba implementados, preparación del entorno de pruebas, número de casos de prueba ejecutados, casos de prueba exitosos/fallidos
  - Métricas de calidad del producto: disponibilidad, tiempo de respuesta, etc.
  - Métricas de defectos: número de defectos por prioridad, defectos encontrados/corregidos
  - Métricas de cobertura: porcentaje de cubrimiento de requerimientos, de código

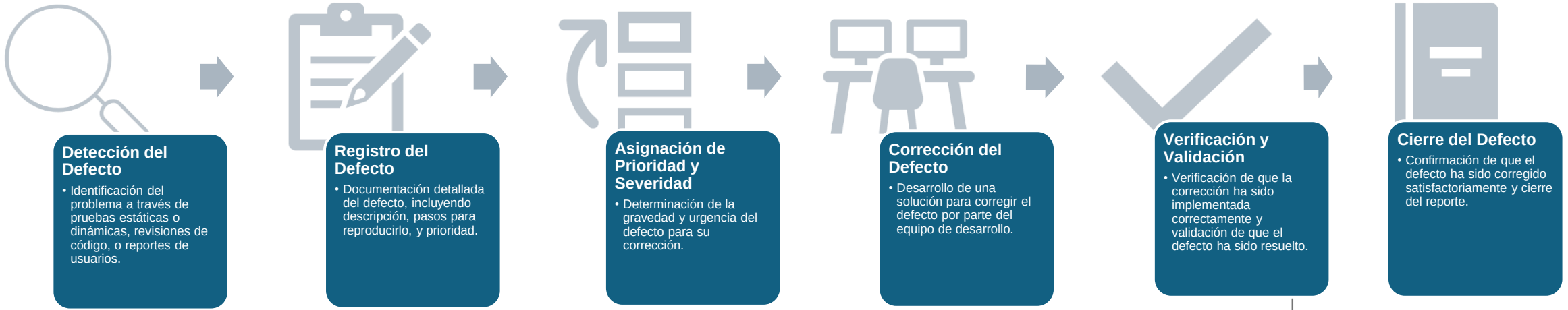
El proceso de medición

Pasos clave:

1. Elegir las medidas a realizar
2. Seleccionar componentes a evaluar
3. Medir características de los componentes
4. Identificar las medidas anómalas
5. Analizar los componentes anómalos

Gestión de Defectos

- Recordando que un defecto es cualquier desviación del comportamiento esperado en el software, que puede causar un mal funcionamiento o un resultado incorrecto.
- Proceso de Gestión de Defectos:



Puntos importantes en la Gestión de Defectos

- Claridad en la Descripción: debe detallarse el defecto de manera precisa, incluyendo pasos para reproducirlo y contexto relevante.
- Asignación Correcta de Prioridad y Severidad: priorizar los defectos en función de su impacto en el usuario y la urgencia de su corrección.
- Comunicación Efectiva: informar a los interesados sobre el estado y la resolución de los defectos de manera clara y oportuna.
- Documentación Completa: registrar toda la información relevante sobre el defecto, incluyendo versiones afectadas, ambiente de prueba y estado actual.
- Seguimiento y Actualización: Realizar un seguimiento regular del estado de los defectos, actualizando su información según sea necesario.
- Validación Post-Corrección: Verificar que la corrección del defecto ha sido implementada correctamente y no ha introducido nuevos problemas.
- Aprendizaje Continuo: recordar que la gestión de defectos es una oportunidad para identificar áreas de mejora en el proceso de desarrollo.


UNIVERSIDAD ORT
Uruguay
Facultad de
Ingeniería
Analizando la Calidad

Análisis causal

- Actividad fundamental donde se abordan las causas de los problemas para erradicar los defectos que se originan de ellos, proponiendo cambios en las herramientas o procesos.
- Se debe realizar en forma temprana para aprovechar la experiencia.
- Es necesario realizar un plan de acciones y seguimiento de los cambios propuestos, es importante el compromiso del gerente.

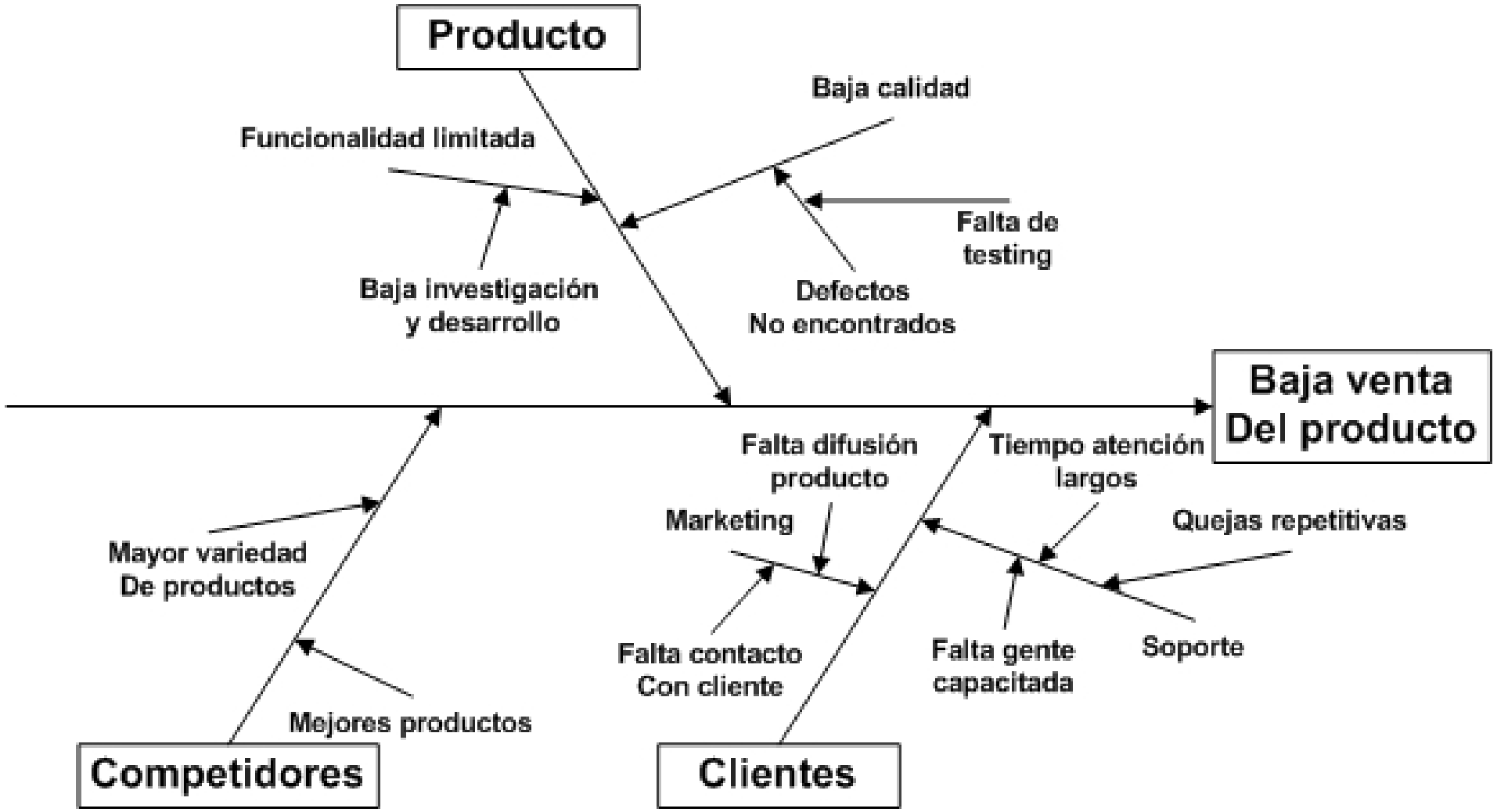
Análisis causal

- Objetivo: prevenir los defectos en base a la eliminación de las causas que los provocan
- Se analizan los registros de defectos detectados
  - registros de inspecciones
  - registros de pruebas
  - registros de reportes de problemas
- Categorizar las causas de los defectos (el 20% de las causas provocan el 80% de los defectos)

Análisis causal

- 5 “por qué”
  - Baja venta del producto
    - ¿Por qué? Porque estamos vendiendo menos
    - ¿Por qué? Porque el cliente no quiere nuestros productos
    - ¿Por qué? Porque el competidor tiene mejores productos
    - ¿Por qué? Porque no hemos producido buenos productos últimamente
    - ¿Por qué? Porque hemos reducido el presupuesto de investigación y desarrollo

Análisis causal – Diagrama de Ishikawa



Facultad de
Ingeniería

Análisis Causal de Defectos

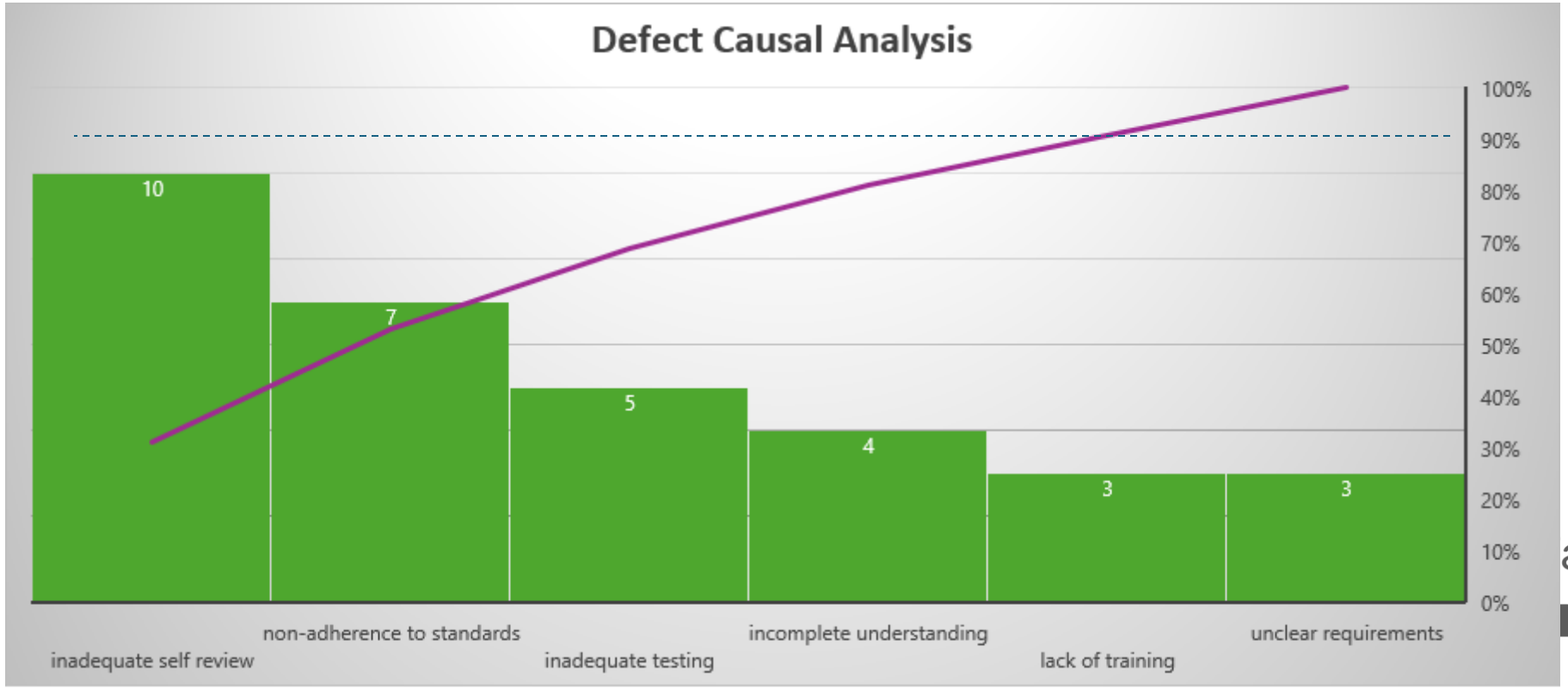
Categoría de Causa	Causas
Definición defectuosa de Requerimientos	• Descripción incorrecta del requerimiento • Omisión de Requerimientos vitales • Definición incompleta de requerimientos
Fallas en la comunicación entre cliente y desarrolladores	• Malos entendidos en las descripciones del requerimiento proporcionadas por el cliente. • Malos entendidos al interpretar las solicitudes de cambio del cliente • Malos entendidos sobre las aclaraciones del cliente a las preguntas del desarrollador
Desviaciones de las especificaciones	• En general, causados por reutilización de módulos/software de productos/proyectos anteriores. • Omisión de funciones específicas debido a restricciones de tiempos • Introducción de cambios al software, por parte del desarrollador sin aprobación/verificación específica.

Análisis Causal de Defectos (cont)

Categoría de Causa	Causas
Errores de diseño lógico	• Algoritmos incorrectos • Definiciones de procesos con errores de secuenciación • Errores en definición de condiciones de borde
Errores de codificación	• Uso de funciones, variables, paquetes, etc
No conformidad con estándares de codificación o documentación	• Falla en el seguimiento de estándares para la codificación • Falla en el seguimiento de estándares para la documentación
Limitaciones en el proceso de pruebas	• Planes de prueba incompletos • Fallas al documentar y reportar defectos • Fallas al no corregir prontamente los defectos reportados.
Errores de Documentación	• Documentación incompleta u obsoleta

Análisis de Pareto

- Utilizado para identificar el 20% de las causas que ocasionan el 80% de los defectos.



acultad de
ngeniería

Bibliografía

- Certified Tester - Foundation Level Syllabus v4,0, International Software Testing Qualifications Board
- R. Pressman. Ingeniería de Software: Un enfoque práctico, McGraw-Hill Interamericana de España S.L.; 9th edition (April 12, 2021)
- I. Sommerville. Software Engineering, 10th ed. Boston, MA: Pearson, 2015
- León Perdomo, Yeniset, Enrique Góngora Rodríguez, Asnier, & Febles Estrada, Ailyn. (2013). **Aplicando métricas de calidad a proyectos y procesos durante las pruebas exploratorias.** *Revista Cubana de Ciencias Informáticas*, *7*(2), 193-205. Recuperado en 17 de junio de 2018, de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992013000200008&lng=es&tlng=es.
- https://iso25000.com/index.php/normas-iso-25000/iso-25010
- Software Quality Assurance - Claude Y. Laporte, Alain April, Published by Wiley-IEEE Computer Society Press