UNIVERSIDAD DE LIMA | 60 años

ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE

UNIDAD 4: PRUEBAS DE SOFTWARE

Temario

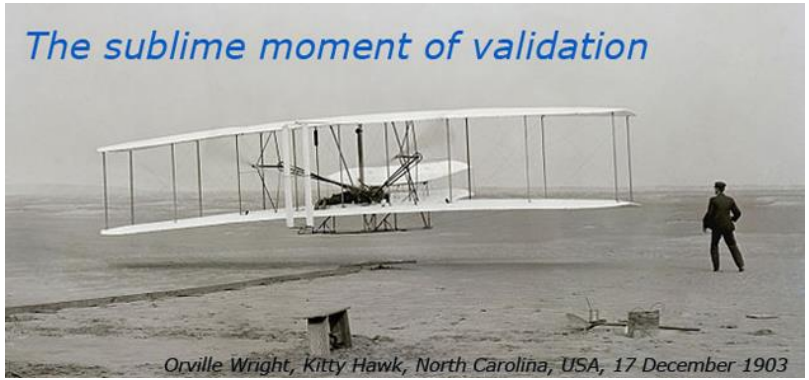
- Validación y verificación (V&V)
- Conceptos básicos de pruebas.
- Tipos de pruebas: Pruebas de funcionalidad, Pruebas de usuario, Pruebas de performance y carga, Pruebas de seguridad.
- Pruebas de caja blanca y Negra.
- Diseño de casos de prueba.
- Diseño de Plan de pruebas.

Validación y verificación (V&V)

Son procedimientos independientes que se usan juntos para verificar que un producto, servicio o sistema cumple con los requisitos y especificaciones y que cumple con su propósito previsto

Verification and Validation (V&V)

- Procesos que aseguran que el software desarrollado **satisface las especificaciones y funcionalidades (hechas para su propósito)** solicitadas por el usuario.
- El nivel de confianza depende de:
  - Las **funciones del software**
  - Las **expectativas del usuario**
  - Y el **entorno del mercado**



UNIVERSIDAD DE LIMA

Verification and Validation (V&V)

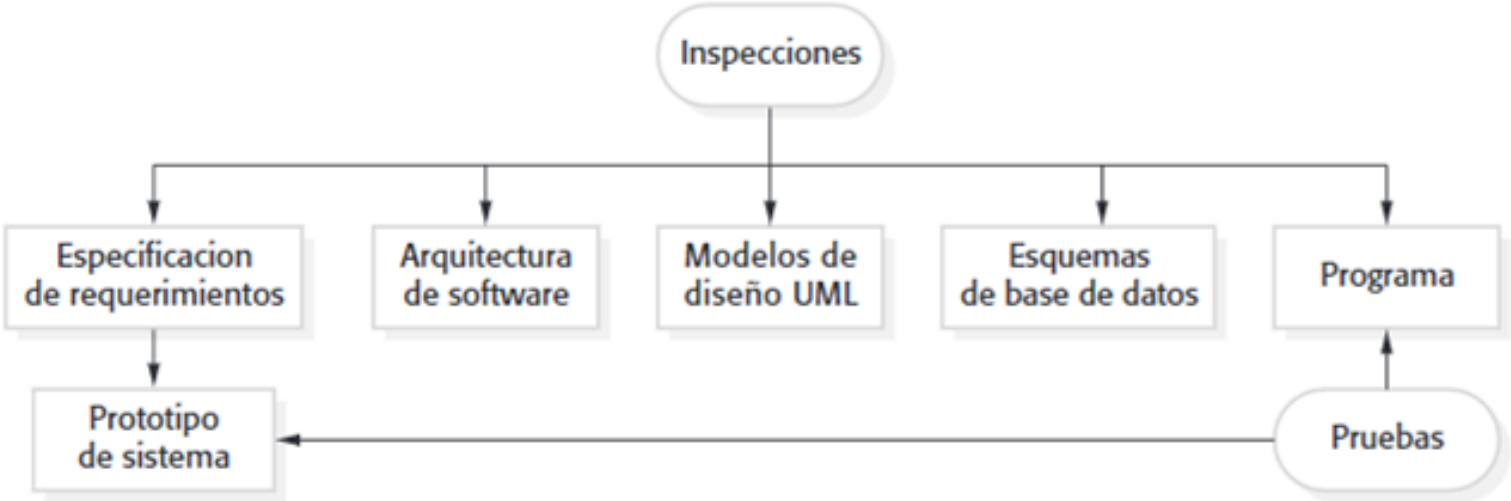
- **Verificación**
  - Comprobar que el software está de acuerdo con su especificación (RF, RNF).
  - **¿Estamos construyendo el producto correctamente?**
    - Salidas esperadas
    - Flujos de información correctas
    - Existencia de Bugs
    - Es más formal
- **Validación**
  - Asegurar que el Software satisface las expectativas del cliente. Va más allá de la comprobación de las especificaciones del Software
  - **¿Estamos construyendo el producto correcto?**
    - Corresponden las especificaciones
    - Cumplir cada requerimiento
    - Expectativas del cliente

Verification and Validation (V&V)

- **Inspecciones de Software**
  - Analizan y comprueban los artefactos del software. Puede usarse las inspecciones en todas las etapas del proceso.
    - Complementadas con análisis automático de código fuente o documentos asociados
    - Las inspecciones y su complemento de análisis automáticos son **técnicas estáticas de V&V** que no necesitan ejecutar el Software
- **Pruebas de Software**
  - Implica ejecutar una implementación del software con datos de prueba
  - Las pruebas son una **técnica dinámica de V&V**

Verification and Validation (V&V)

- ¿Cuándo hacer inspecciones y pruebas?



UNIVERSIDAD DE LIMA

UNIVERSIDAD DE LIMA

60 años

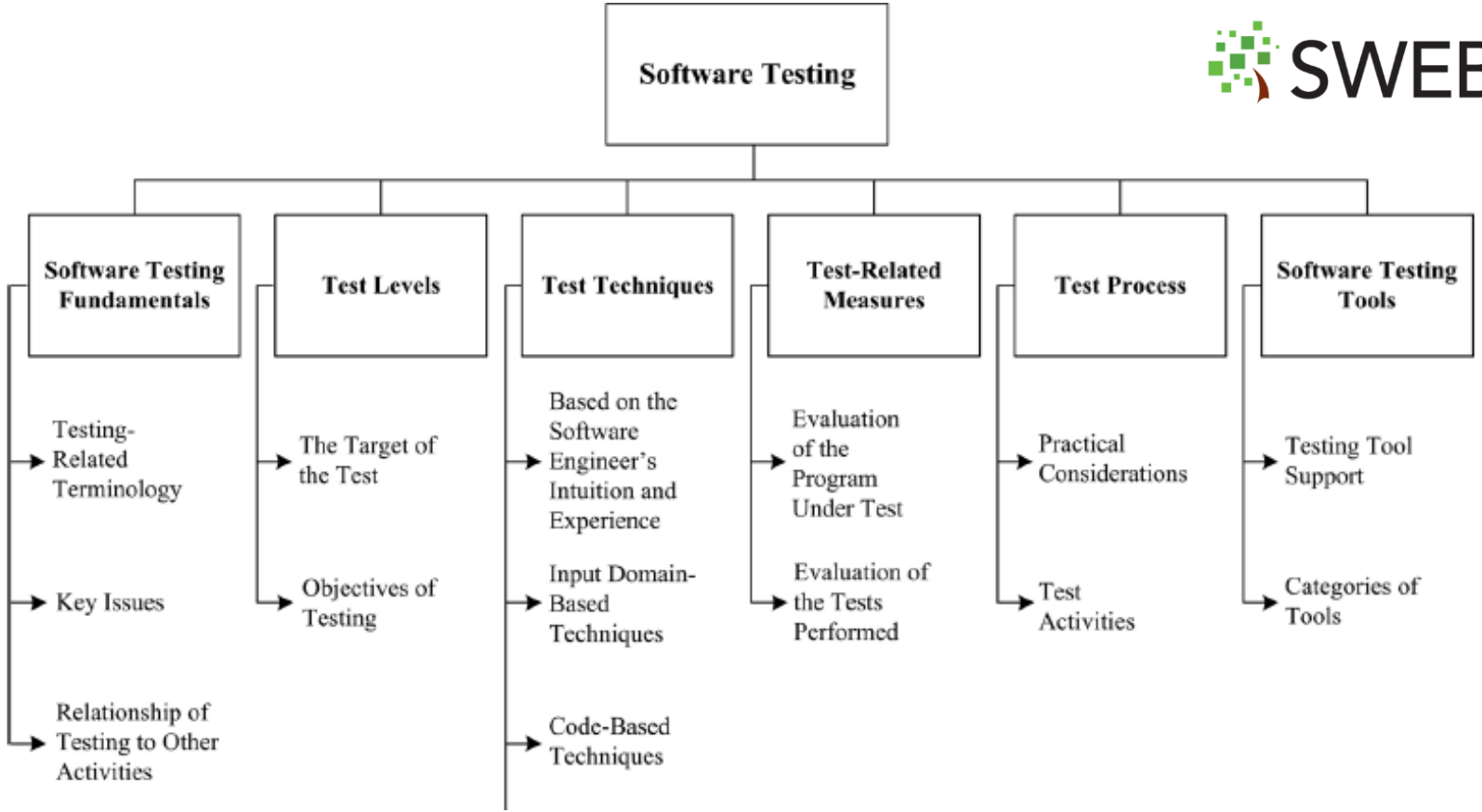
Conceptos básicos de pruebas

PRUEBAS DE SOFTWARE

- ¿Qué es una prueba de Software?
  - Proceso que busca demostrar que el sistema hace lo que debe hacer, localizando errores y/o anomalías.

UNIVERSIDAD DE LIMA

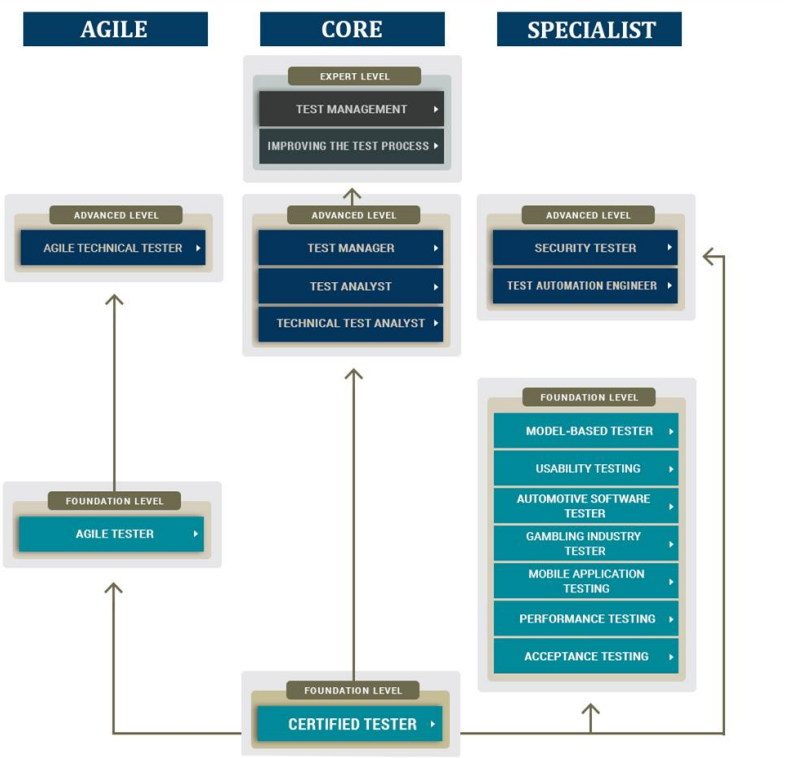
Software Testing según SWEBOK V3.0



SWEBOK V3.0

RSIDAD
A

Certifying Software Testers Worldwide



ISTQB
International Software Testing Qualifications Board

https://www.istqb.org/

UNIVERSIDAD DE LIMA

“Las pruebas sólo pueden mostrar la presencia de errores, mas no su ausencia.”

(Dijkstra et al., 1972)

¿Por qué hacer pruebas de Software?

Learning from Pokémon GO: The Importance of Testing

https://spin.atomicobject.com/2016/08/03/pokemon-go-app-testing/

¿Cómo ha manejado Cineplanet su última crisis digital?

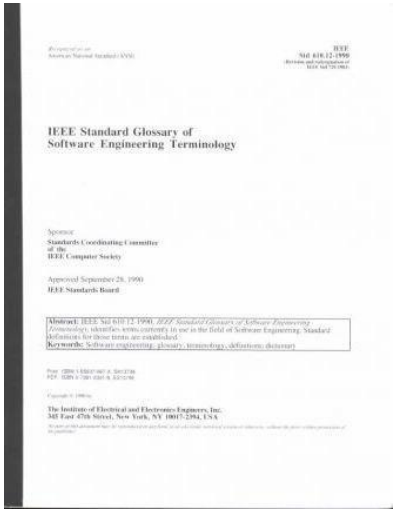
https://elcomercio.pe/economia/negocios/cineplanet-resolvio-crisis-digital-noticia-492206-noticia/

UNIVERSIDAD DE LIMA

Conceptos Generales

- **Defecto (defect)**
  - Es el resultado de una deficiencia o error durante la construcción del software.
  - Elemento fuera de rangos normales
  - Pertenece a algo más grande
  - No corresponde a sus especificaciones
- **Fallo (failure)**
  - Es la incapacidad del software (o una de sus partes) de realizar sus funciones específicas.
  - Se ha ocasionado un resultado no esperado
  - Hay (¡o no hay!) un mensaje de error
  - Una interrupción

610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology


IEEE Standard Glossary of Software Engineering Terminology

Sponsor
Standards Coordinating Committee of the IEEE Computer Society

Approved September 26, 1990
IEEE Standards Board

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

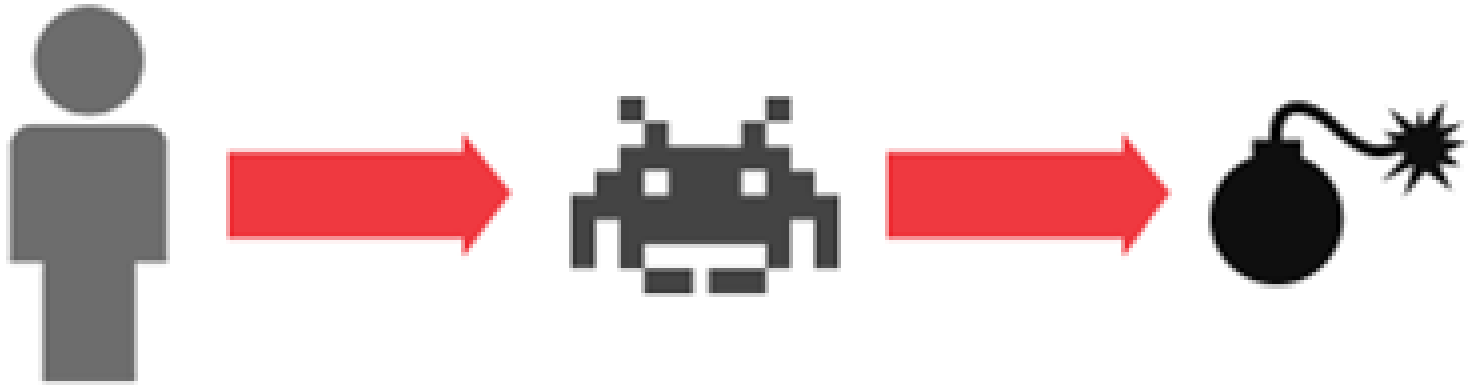
UNIVERSIDAD DE LIMA

Conceptos Generales

- **Error (bug, fault)**
  - Una equivocación cometida por el desarrollador.
  - Pueden ser:
    - De requisitos
    - De diseño
    - De implementación

UNIVERSIDAD DE LIMA

Relación entre Error, Defecto y Fallo



ERROR
del programador

DEFECTO
en el software

FALLO
en el sistema

UNIVERSIDAD DE LIMA

Conceptos Generales

- **Errores comunes**
  - División entre cero
  - Ciclo infinito
  - Problemas aritméticos como desbordamientos (overflow)
  - Condición de carrera
  - Exceder el tamaño del arreglo
  - Utilizar una variable no inicializada
  - Acceder a memoria no permitida (Violación de acceso)
  - Pérdida de memoria (memory leak)
  - Desbordamiento o subdesbordamiento del arreglo
  - Desbordamiento de búfer (buffer overflow)
  - Bloqueo mutuo (deadlock)
  - Indizado inadecuado de tablas en bases de datos

Conceptos Generales

- **¿Quién debe hacer las pruebas?**
  - El equipo de desarrollo
    - Conoce cómo debe ser el funcionamiento del sistema.
    - Puede tener un sesgo al probar (evita comportamientos raros)
    - Puede ser más económico (a corto plazo)
  - Un equipo sólo de pruebas
    - Personas especializadas en calidad
    - Es más costoso
    - No tienen sesgos al probar (más objetivos)
    - Paralelizable con desarrollo

UNIVERSIDAD DE LIMA

Conceptos Generales

- **Depuración (debugging)**
  - Proceso para corregir los errores y problemas encontrados por las pruebas.
    - Eliminar los bugs encontrados.
    - Usar las salidas de las pruebas (mensajes, resultados esperados, datos introducidos) para brindar soluciones.
    - Realizadas por el equipo de desarrollo.
    - Se debe dar un tiempo estimado (explícito) en el cronograma.

UNIVERSIDAD DE LIMA

Conceptos Generales

- **Pase a producción**
  - Proceso de instalación, implantación o puesta a disposición de un Software en el ambiente donde debe realizar sus funciones.
    - Culminadas las pruebas.
    - Pasar del entorno de desarrollo al entorno final.
      - Pueden haber entornos de pre-producción intermedios.
    - Dependiendo de la complejidad del sistema, puede ser un proceso largo o muy directo.

UNIVERSIDAD DE LIMA

UNIVERSIDAD DE LIMA | 60 años

Tipos de pruebas

TIPOS DE PRUEBAS

- Podemos clasificarlas por granularidad, por el método para aplicarlas, por las técnicas que se usan, por los objetivos que se buscan, etc.

UNIVERSIDAD DE LIMA

TIPOS DE PRUEBAS

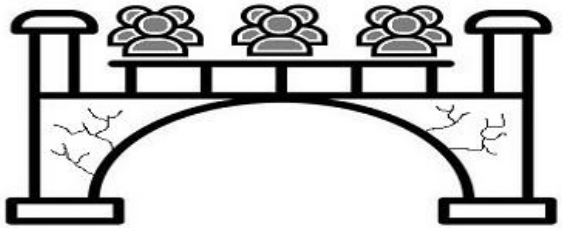
UNIVERSIDAD DE LIMA

TIPOS DE PRUEBAS

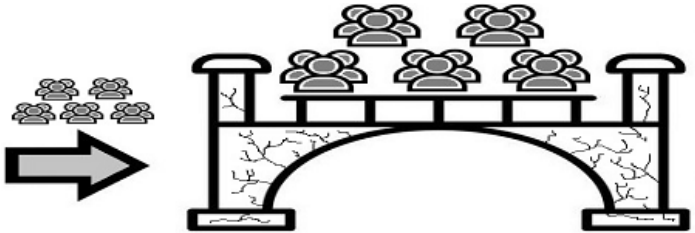
- **Pruebas de desarrollo**
  - Todas las actividades de pruebas en la elaboración del Software.
  - **Clasificación por granularidad** incluyen:
    - Pruebas de Unidad
    - Pruebas de Componente
    - Pruebas de Sistema

TIPOS DE PRUEBAS

- **Pruebas de componente**
  - Se enfocan en el uso de las interfaces entre los componentes. Además, se pone a prueba la arquitectura del software.
  - Pueden detectar:
    - Uso incorrecto de interfaz
    - Mala interpretación de la interfaz
    - Errores temporización (carrera)
  - Se pueden usar **pruebas de estrés** en esta etapa.



Prueba de Carga



Prueba de Estrés

UNIVERSIDAD DE LIMA

TIPOS DE PRUEBAS

- **Pruebas de componente**
  - Lineamientos
    - Listar llamados de interfaces
    - Forzar llamados a nulos (si no está el componente)
    - Probar que pasa cuando falla el otro componente
    - Pruebas de esfuerzo al enviar mensajes
    - Pruebas de memoria compartida

TIPOS DE PRUEBAS

- **Pruebas del Sistema**
  - Se prueba el Sistema por Completo
  - Se dan las pruebas de integración completa
  - Pruebas usando los casos de uso/actividades:
    - -Colocar todas las entradas, incorrectas o correctas
    - -Seguir el proceso completo
    - -Probar todas las combinaciones

TIPOS DE PRUEBAS

- **Prueba de Integración**
  - Las pruebas de integración son parte de las pruebas de sistema. Para integrar, se pueden usar diferentes técnicas:
    - TopDown
    - BottomUp
    - Sandwich
    - BigBang

TIPOS DE PRUEBAS

- **Prueba de Versión**
  - Pruebas que se pueden realizar para cada release (lanzamiento) de un sistema.
    - Pruebas basadas en requerimientos
    - Pruebas de escenarios
    - Pruebas de rendimiento
    - Pruebas de regresión
    - Pruebas de aseguramiento de calidad de datos
  - Dependen mucho del tipo de sistema y proyecto.

TIPOS DE PRUEBAS

- **Pruebas basadas en requerimientos**
  - Analizamos cada requerimiento y se diseña una prueba por cada uno.
  - Los requerimientos deben ser medibles y específicos.
- **Pruebas de escenarios**
  - Generar escenarios según roles en el sistema.
  - Especificar interacciones con el usuario.
- **Pruebas de aseguramiento de calidad de datos**
  - Revisar que los datos corresponden a lo esperado.
  - Puede ser para nuevos sistemas o migraciones.

TIPOS DE PRUEBAS

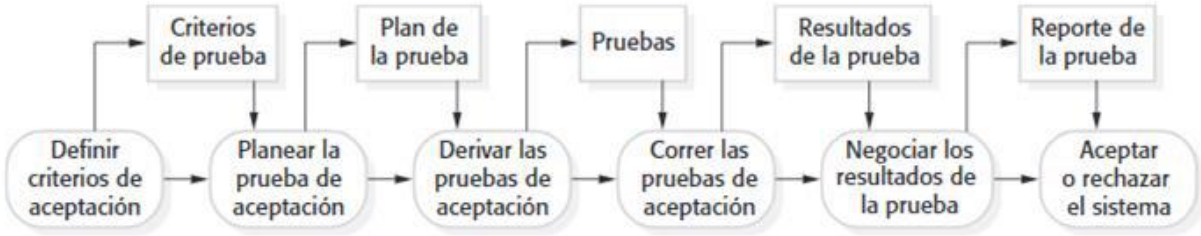
- **Pruebas rendimiento**
  - Prueba de esfuerzo (estrés)
    - Necesariamente requiere automatización. Ej: Jmeter, AB, Siege.
    - Se debe aguantar más de lo máximo necesario.
  - Prueba de seguridad
    - Se sigue un protocolo para probar capas (red) del sistema.
- **Pruebas de regresión**
  - Evaluar que el cambio de versión no rompe funcionalidad anterior.
  - Puede incluir otras pruebas.

TIPOS DE PRUEBAS

UNIVERSIDAD DE LIMA

TIPOS DE PRUEBAS

- **Pruebas de Usuario**
  - Cuando el usuario (cliente) apoya en las pruebas del sistema.
    - **Pruebas Alfa:** Se trabaja con un sistema que aún está en desarrollo, junto con el equipo.
    - **Pruebas Beta:** Se permite el uso libre de usuarios al sistema para que experimenten.
      - "Always in Beta"
      - Release Candidate
    - **Pruebas de Aceptación:** Se decide si el sistema ya pasa a producción. En XP (y otras metodologías ágiles), las pruebas de aceptación son parte del desarrollo del sistema, por lo que no existe como paso separado.



Elaboración de Pruebas de Aceptación

- **¿Qué son las pruebas de aceptación?**
  - También se les conoce como pruebas de usuario.
  - Nos permiten determinar si lo que estamos construyendo cumple con los requerimientos de nuestros usuarios.
  - Normalmente se manejan como pruebas de caja negra (punto de vista de usuario), abstrayéndose de todo detalle de implementación.
  - Normalmente al cumplir con estas pruebas, se considera que el software es aceptado por nuestros usuarios.

Elaboración de Pruebas de Aceptación

- **Tipo**

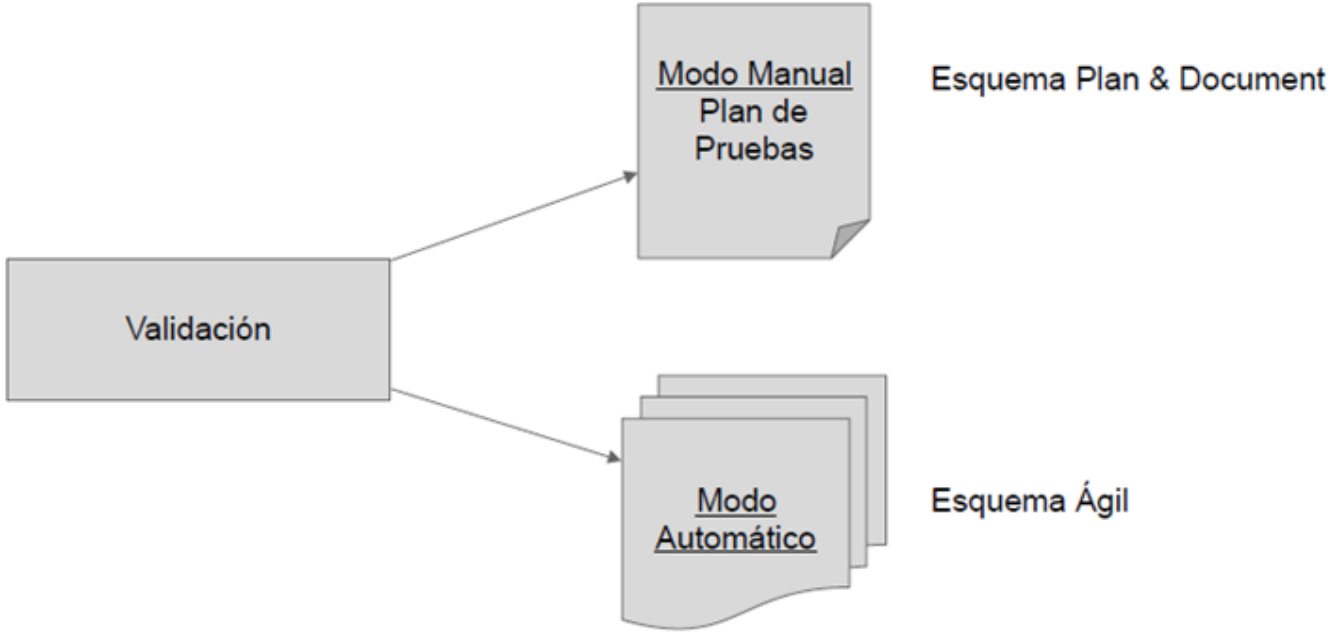
Funcional

Usabilidad

UNIVERSIDAD DE LIMA

Elaboración de Pruebas de Aceptación

- **Estrategia para pruebas de aceptación**



UNIVERSIDAD DE LIMA

Elaboración de Pruebas de Aceptación

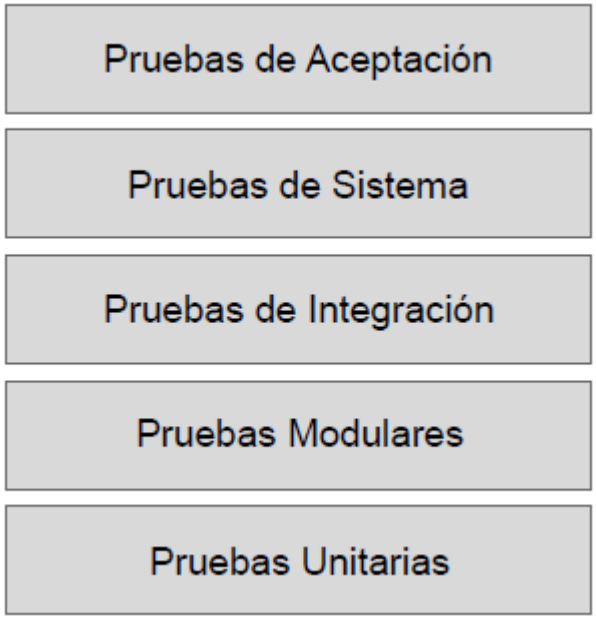
- **Plan de Pruebas Funcionales**
  - Bajo el enfoque Plan & Document, al iniciar el proyecto (de preferencia) se deberá de implementar un documento llamado Plan de Pruebas que cuente:
    - Casos de pruebas con sus escenarios (instrucciones para realizar la prueba y criterios de aceptación).
    - Escenarios no contemplados en las pruebas.
    - Riesgos
    - Participantes (tanto internos como externos).

Elaboración de Pruebas de Aceptación

- **Plan de Pruebas de Usabilidad**
  - Se trata de evaluar el comportamiento de los usuarios durante el uso del producto de software.
  - Usualmente durante los tests se pide que ciertos usuarios realizan tareas comunes con el software para que así los miembros del equipo puedan hacer notas del comportamiento de estos.
  - Objetivos:
    - Descubrir problemas de usabilidad.
    - Obtener data cuantitativa y cualitativa que nos permitan valorar la satisfacción de nuestros usuarios.

Estrategia para las pruebas según el nivel de las actividades

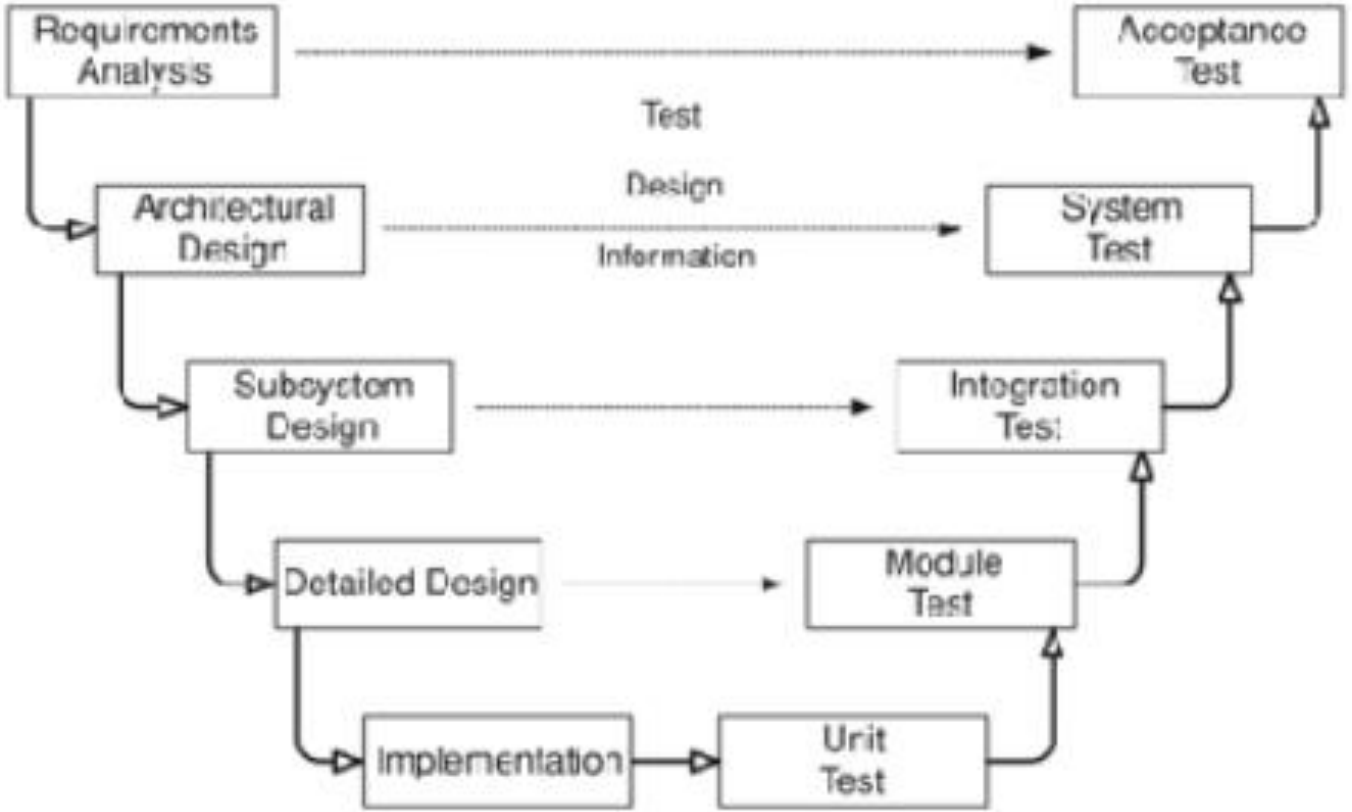
- Realizaremos pruebas en diferentes etapas del desarrollo de software.
- Comenzaremos a un nivel bastante alto y progresivamente iremos descendiendo en nuestros niveles de abstracción.



Actividades de Sofware vs Niveles de Testing (Modelo V)

- Requerimientos (features) del producto.
- Pruebas de la comunicación entre los distintos componentes del sistema (componentes de más alto nivel).
- Pruebas de la comunicación entre los distintos módulos (clases).
- Pruebas de los módulos (clases).
- Pruebas de las funciones.

Actividades de Sofware vs Niveles de Testing (Modelo V)



UNIVERSIDAD DE LIMA

UNIVERSIDAD DE LIMA | 60 años

Pruebas de caja blanca VS pruebas de caja negra

Pruebas de caja blanca

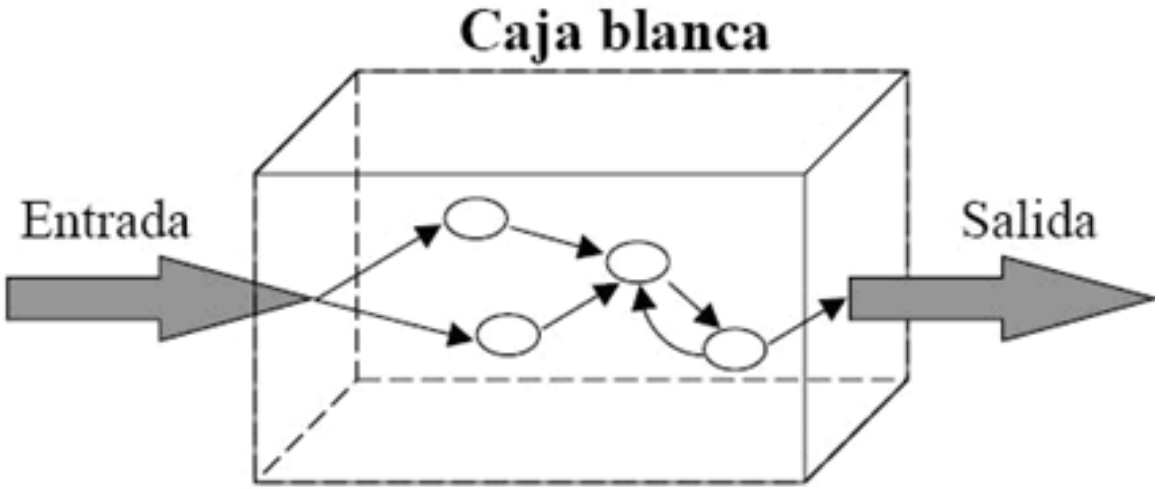
- Pruebas de caja blanca (White-Box Testing). Son pruebas estructurales. Conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones. Ejemplos típicos de ello son las pruebas unitarias. Se centran en lo que hay codificado o diseñado a bajo nivel por lo que no es necesario conocer la especificación de requisitos, que por otra parte será difícil de relacionar con partes diseñadas a muy bajo nivel.

Pruebas de caja negra

- Las pruebas de caja negra (Black-Box Testing) son pruebas funcionales. Se parte de los requisitos funcionales, a muy alto nivel, para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer como está construido por dentro (Caja negra). Las pruebas se aplican sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que se producen para determinar si la función se está desempeñando correctamente por el sistema bajo prueba. Las herramientas básicas son observar la funcionalidad y contrastar con la especificación.

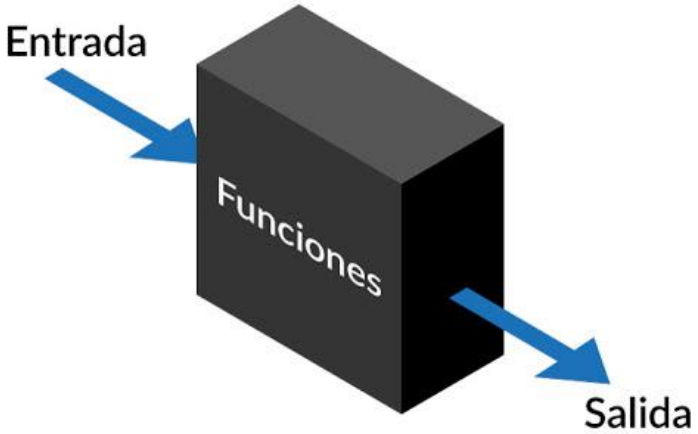
TIPOS DE PRUEBAS

- **Pruebas de Caja Blanca**
  - Se analiza como el procedimiento va pasando parte por parte
    - Dependiendo de la granularidad, se puede hacer a distintos niveles
    - Pruebas de cobertura (¿Se pasa por todo el código?)



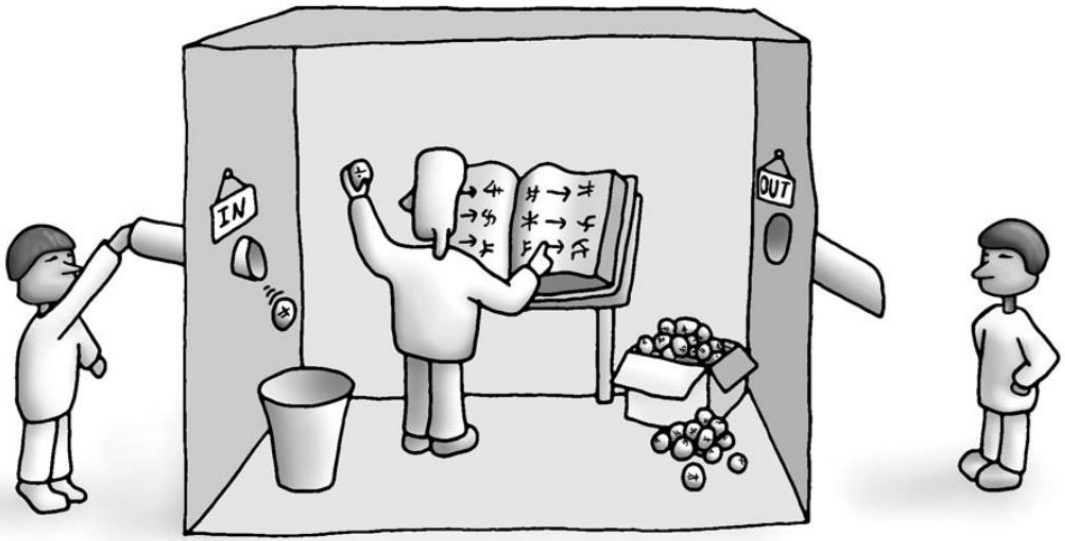
TIPOS DE PRUEBAS

- **Prueba de Caja Negra**
  - Sólo enfocarse en entradas vs salidas esperadas
  - Se ve el componente/módulo sin el detalles interno.
  - Se usan las clases de equivalencia



UNIVERSIDAD DE LIMA

Prueba de Caja Blanca VS Caja Negra



jolyon.co.uk

UNIVERSIDAD DE LIMA

UNIVERSIDAD DE LIMA | 60 años

Diseño de casos de prueba

Para definir las pruebas a realizar se debe tener en consideración: Costo, Tiempo y Recursos; la calidad no debería ser negociable; Clases de Equivalencia & Otras Técnicas

Casos de pruebas

- Se deben elegir casos efectivos, es decir:
  - Mostrar que el sistema funciona como se esperaba cuando se usa como se indica.
  - Si hay “errores”, que sean visibles.
- Es decir 2 tipos de casos de pruebas:
  - Caso(s) ideal(es)
  - Caso(s) con problemas comunes

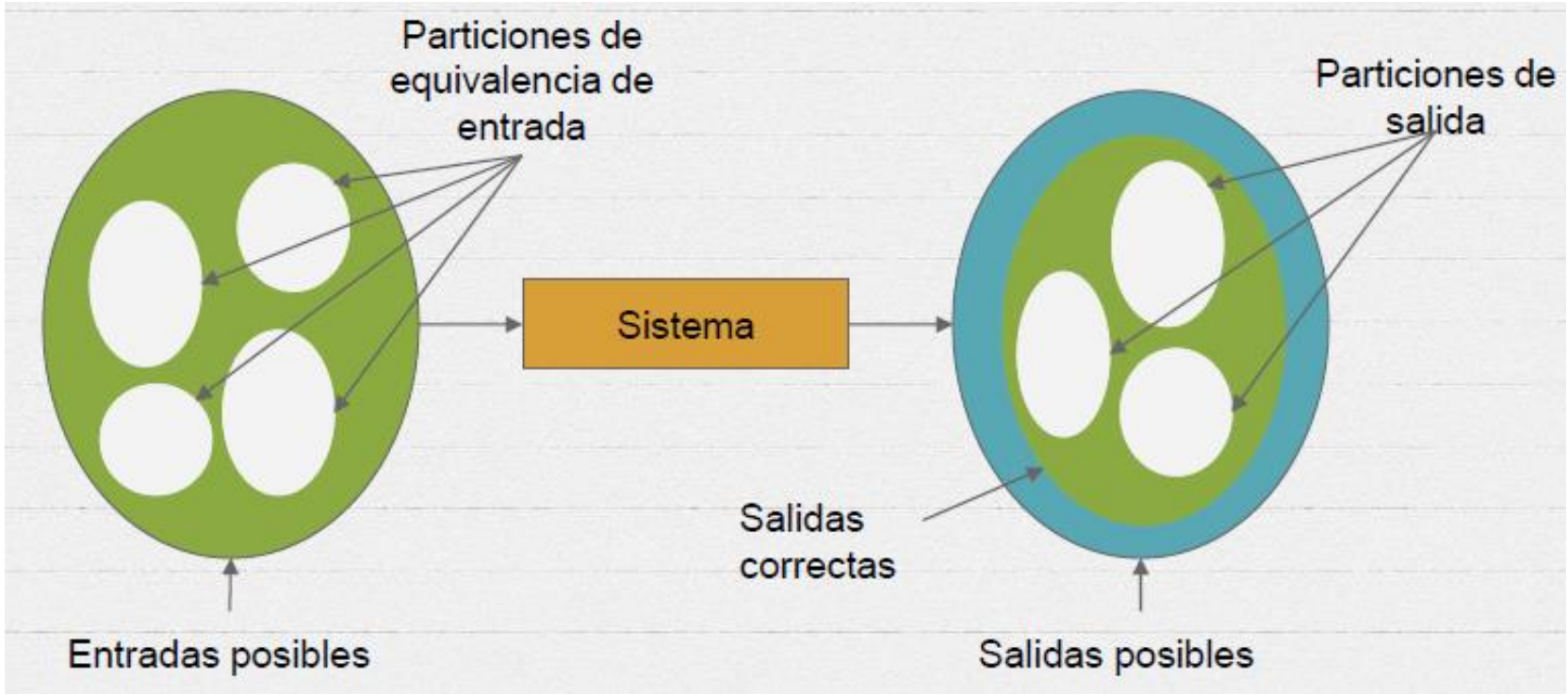
Casos de pruebas

- Dos posibles estrategias son:
  - Pruebas de partición (Clases de equivalencia)
    - Se definen grupos de datos de entrada
    - Se usan puntos limítrofes (valores límite)
  - Pruebas basadas en lineamientos
    - Basadas en la experiencia de errores

Pruebas de partición

- Normalmente los datos de entrada pueden agruparse por características similares.
  - Estos grupos se llaman “clases de equivalencia”.
  - Estas clases se usan para definir (cada una) un caso de prueba.
  - Los valores límites deberían probarse por separado para ver comportamientos no esperados. (Pruebas de valor límite).
  - Se tipifican como pruebas de caja negra.

Clases de equivalencia

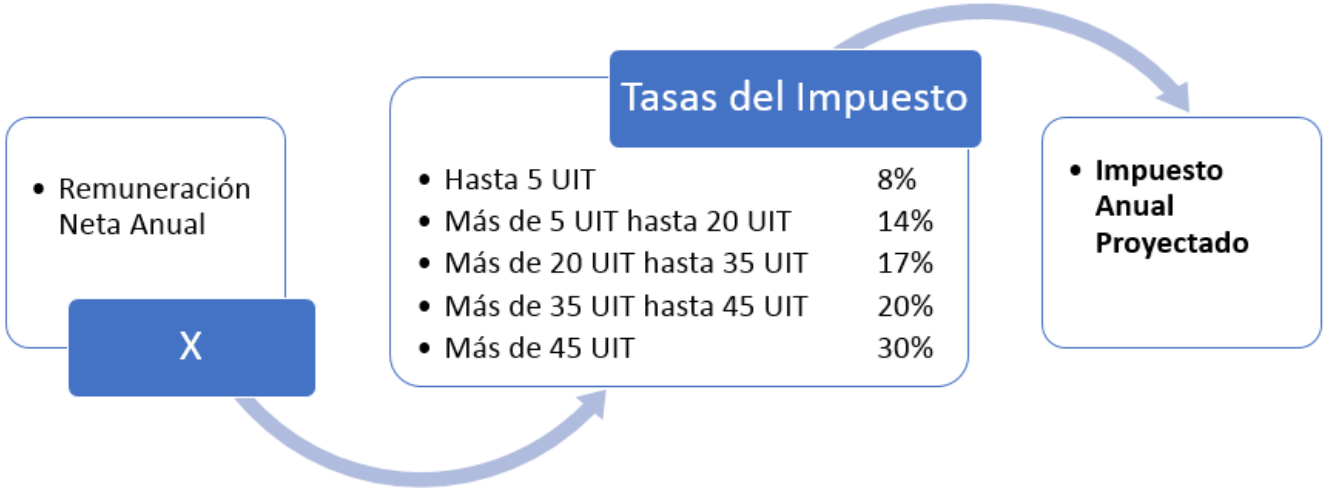


Clases de Equivalencia

- Si se especifica un rango de valores, se creará una clase válida y dos clases no válidas.
- Si se sospecha que ciertos elementos de una clase no se tratan igual que el resto de la misma, deben dividirse en clases menores.
- Si se especifica un conjunto de valores admitidos y se sabe que el programa trata de forma diferente cada uno de ellos, se identifica una clase válida por cada valor.
- Si se especifica una condición booleana (por ejemplo, “el primer caracter debe ser una letra”), se identifica una clase válida (“es una letra”) y una no válida (“no es una letra”).

Clases de Equivalencia (Ejemplo1)

- Para calcular el Impuesto a la Renta se debe tomar en cuenta que las tasas varían según la Remuneración Neta Anual (RNA).
- Armar las clases de equivalencia que se necesitaría para probar el ingreso como dato de entrada del RNA en un sistema.
  - Nota: UIT en 2018 es S/ 4 300,00 soles
  - Asumir que de todos los sueldos se cobra el IR.
  - Asumir un máximo de 10 millones de soles



Clases de Equivalencia (Ejemplo1)

Condición de Entrada	Clases Válidas	Clases no Válidas
Campo Remuneración Neta Anual	1. Valores mayores a 0 y menor o igual a 20 750	6. Vacío
	2. Valores mayores a 20750 y menor o igual a 83 000	7. Valor menor o igual a 0
	3. Valores mayores a 83 000 y menor o igual a 145 250	8. Valor mayor a 10 millones
	4. Valores mayores a 145 250 y menor o igual a 186 750	9. Datos no numéricos
	5. Valores mayores a 186 750 y menor o igual a 10 millones.	

Clases de Equivalencia (Ejemplo2)

- Para registrarse en un sistema, el usuario debe ingresar una contraseña que cumpla las siguientes condiciones:
  - Debe incluir al menos una mayúscula
  - Debe incluir al menos una minúscula
  - Debe incluir al menos un número
  - Debe tener una longitud mínima de 6 caracteres
  - Debe tener una longitud máxima de 14 caracteres
- Armar las clases de equivalencia que se necesitaría para probar que la contraseña es válida. Asumir sólo caracteres alfanuméricos.

Clases de Equivalencia (Ejemplo2)

Condición de Entrada	Clases Válidas	Clases no Válidas
Campo Contraseña	**1.** Entre 6 y 14 caracteres, con 1 o más mayúsculas, con 1 o más minúsculas, con 1 o más caracteres numéricos	2. Vacío
		3. Sin caracteres en mayúscula
		4. Sin caracteres en minúscula
		5. Sin caracteres numéricos
		6. Entre 1 y 5 caracteres
		7. Más de 14 caracteres

Pruebas basadas en lineamientos

- Se debe tomar en cuenta la experiencia del equipo y de la industria para probar errores recurrentes.
- Algunos generales:
  - Entradas que fuercen todos los mensajes de error
  - Entradas que desborden el buffer
  - Repetir las mismas acciones una y otra vez
  - Forzar salidas inválidas
  - Forzar cálculos muy grandes o muy pequeños

UNIVERSIDAD DE LIMA | 60 años

Diseño de Plan de pruebas

Especificación de un caso de Prueba

- Una vez diseñada la prueba, se debe crear su especificación. Esta debe contener por lo menos:
  - Identificador
  - Objetivo
  - Precondición
  - Descripción de la prueba
  - Resultados esperados
- Ojo: Dependiendo del proyecto o institución, puede requerir otros elementos.

Especificación de un caso de Prueba

Ejemplo

Prueba MU-25	
Objetivo	Dar de alta a un usuario dejando vacío el nombre de usuario.
Precondición	Se ingresó al sistema como “registrador”
Descripción de la prueba	En la interfaz de entrada introducir - Username: usuario1 - Primer Apellido: apellido - Segundo Apellido: apellido - Tipo de usuario: estándar Dejar vacío el campo “Nombre”
Resultados esperados	Se muestra el mensaje “Todos los campos son obligatorios”

Casos de Prueba & Clases de Equivalencia

- Considerar que para cada combinatoria de clases válidas se debe crear un caso de prueba.
- Considerar que para cada clase inválida se debe crear un caso de prueba.

Clases de Equivalencia (Ejemplo2)

Condición de Entrada	Clases Válidas	Clases no Válidas
Campo Código	**1.** Cadena numérica de 5 caracteres	2. Vacío (0 caracteres)
		3. Entre 1 y 4 caracteres
		4. Más de 5 caracteres
		5. Cadena de caracteres no numéricos
Campo Tipo de Usuario	6. Estándar	
	7. Administrativ**o**	

Casos de Prueba Válidos: (1,6) (1,7)
Casos de Prueba Inválidos: (2,6) (3,6) (4,7) (5,7)

UNIVERSIDAD DE LIMA

Plan de Pruebas

- Especifica cómo se deben ejecutar las pruebas.
- Puede incluir:
  - Alcance de las pruebas
  - Calendarización de las actividades
  - Especificación de las características e ítems a probar
  - Identificación de responsables en cada actividad de las pruebas
  - Casos de prueba especificados
  - Riesgos asociados

Documentación de Pruebas

- Dependiendo de la organización y/o proyecto se pueden pedir cierta documentación adicional. Algunos documentos según el estándar IEEE 829*:
  - Plan de pruebas
  - Especificación de diseño de pruebas
  - Especificación de casos de pruebas
  - Procedimiento de pruebas
  - Reporte de transmisión de ítems para pruebas
  - Informe (Log) de pruebas
  - Reporte de incidentes
  - Reporte de pruebas

Ejercicio

- Liste las clases de equivalencia para diseñar los casos de prueba de la pantalla en la siguiente diapositiva, usando las consideraciones indicadas:
  - Todos los campos son obligatorios.
  - El nombre y apellidos deben ser de máximo 30 caracteres c/u.
  - El DNI es la única opción en Tipo de Documento y debe ser numérico de 8 caracteres.
  - El teléfono debe ser numérico de 7 u 8 caracteres.
  - Sólo considerar caracteres alfanuméricos.

UNIVERSIDAD DE LIMA

Referencias

- Proceso de Construcción de Software 2, Mag. Natalí Flores Lafosse, Maestría en Informática de la PUCP, 2018.
- Sommerville, I. (2005). Ingeniería del software. Pearson educación.
- Bourque, P., & Fairley, R. E. (2014). Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.
- Doria, H. G. (2001). Las Metricas de Software y su Uso en la Region.