Definición de V&V

- Boehm
  - Verificación
    - ¿Estamos construyendo el producto correctamente?
  - Validación
    - ¿Estamos construyendo el producto correcto?
- IEEE
  - Verificación
    - The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of the phase
  - Validación
    - The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements

Errores, Faltas y Fallas

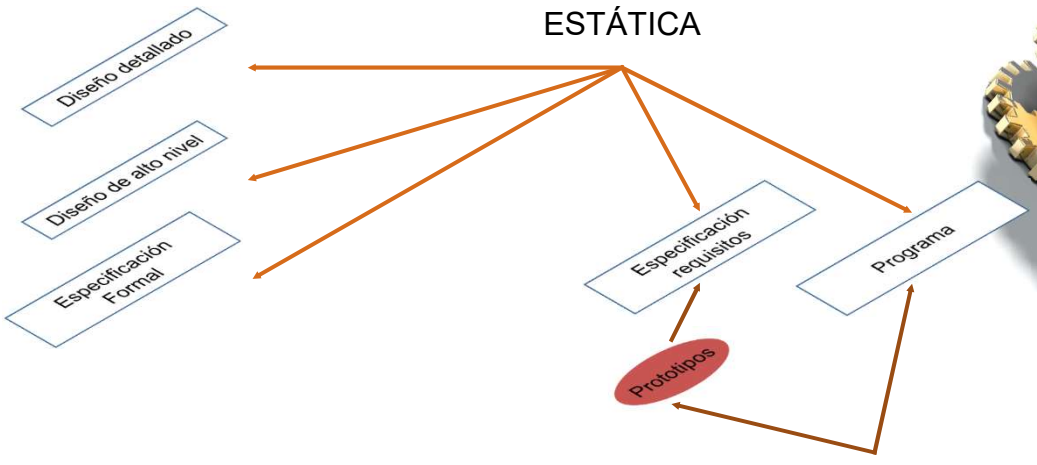
un error humano **puede generar** **una falta (interna)** **que puede generar** **una falla (externa)** **?!**

Tipos de Faltas

- en algoritmos
- de sintaxis
- de precisión y cálculo
- de documentación
- de estrés o sobrecarga
- de capacidad o de borde
- de sincronización o coordinación
- de capacidad de procesamiento o desempeño
- de recuperación
- de estándares y procedimientos
- relativos al hardware o software del sistema

Para que voy a probar? , Que quiero encontrar?
Qué probaré? Existen técnicas / herramientas?
Como? Que necesito para probar? Quien? Cuando?

Técnicas de V&V Estáticas/ Dinámicas



Control de calidad

- Las técnicas de control de calidad buscan defectos en **productos de trabajo del desarrollo de software,** temprana y eficazmente.
- Llevando a cabo
  - inspecciones,
  - walkthroughs estructurado
  - otros métodos de revisiones formales o informales.
  - pruebas

Conformancia & Adecuación

- Uno de los propósitos más importantes de la inspección es la aceptación del producto, es decir, “si el producto cumple conciertos estándares y/o si debe adaptarse”.
- Estas disposiciones incluyen varias decisiones importantes:
  - Conformancia: Juzgar si el producto se desarrollo de acuerdo a las especificaciones
  - Adecuación para el uso: Decidir si un producto es adecuado para el uso

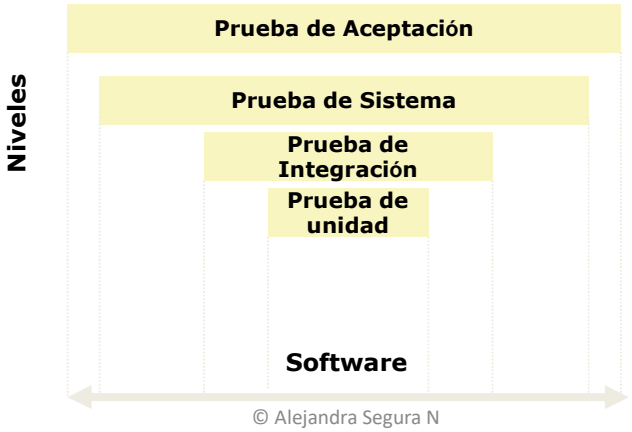
Fundamentos de la Prueba

- Prueba de software. Es la acción de establecer la existencia de errores en el software a través de la ejecución de sus componentes.
- Principios de la Prueba.
  - Una prueba tiene éxito si descubre un error no detectado hasta entonces.
  - La prueba no puede asegurar la ausencia de defectos, **sólo puede demostrar que existen defectos en el software.**

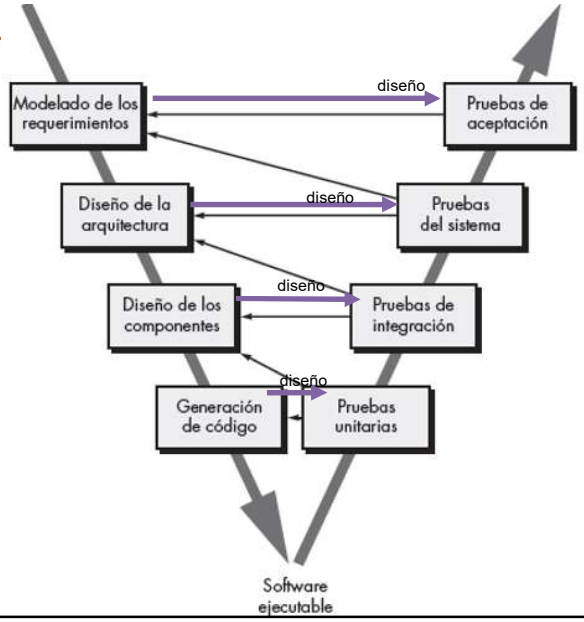
Conceptos

- **Casos de prueba**. Es una especificación de entrada, una descripción de la función del sistema a ejercitarse por esas entradas y una declaración de las salidas esperadas.
- **Técnica de prueba**. Orientan al desarrollador para derivar casos de prueba.

Proceso de Prueba



Modelo V



Prueba Unidad

- Esta etapa se concentra en la prueba individual de unidades de programación. Estas unidades algunas veces son llamadas *módulos* o *módulos atómicos* y representan la entidad de programación más pequeña.
  - Evalúa si la unidad cumple con sus especificaciones

Pruebas Integración

- Objetivo: "intenta conocer cómo funcionarán en conjunto los módulos que anteriormente fueron probados como unidades"
- Se basa en una estructura de programa; arquitectura del Software definida en el diseño.

Prueba Sistema

- En esta etapa se busca probar que el software funciona de acuerdo a las expectativas del cliente, es decir, especificaciones de requerimientos del software. Considerando:
  - restricciones organizacionales
  - requerimientos de calidad
  - e interfaces hw, Software, redes y telecomunicaciones

Prueba Aceptación

- Este tipo de prueba involucra el examen de todo el sistema computacional, es decir el software con todos sus componentes sujetos al hardware y otras interfaces, realizada con los usuarios y con el cliente o un representante de él, a través de la cual se da término al desarrollo y paso a la implantación

Pruebas en el Proceso

- Módulo, Componente o unitaria
  - Verifica las funciones de los componentes
- Integración
  - Verifica que los componentes trabajan juntos como un sistema integrado
- Aceptación
  - Bajo la supervisión del cliente, verificar si el sistema cumple con los requerimientos del cliente (y lo satisface)
  - Validación del sistema

Pruebas en el Proceso

- Desempeño
  - Determina si el sistema integrado, en el ambiente objetivo cumple los requerimientos de tiempo de respuesta, capacidad de proceso y volúmenes
- Estrés
  - probar los límites que un sistema puede soportar.
- Funcional
  - Determina si el sistema integrado cumple las funciones de acuerdo a los requerimientos
- Instalación
  - El sistema queda instalado en el ambiente de trabajo del cliente y funciona correctamente

Enfoques Pruebas de unidad

- Las técnicas que derivan casos de prueba sin referencia a la construcción del programa (ellos fueron creadas como referencia a la especificación o alguna otra descripción de que debería hacer el SW) fueron llamadas “técnicas de caja negra”
  - También denominadas como técnicas funcionales ó “basadas en la especificación”

Enfoques caja negra

- Enfoque caja negra permite encontrar las siguientes categorías de errores:
  - Funciones incorrectas o ausentes
  - Entradas y Salidas erróneas, incompletas o no válidas
  - Errores de interfaz
  - Errores de estructuras de datos o en acceso a BD
  - Errores de rendimiento

Técnicas Enfoque Caja Negra

- **Particionamiento equivalente** busca dividir los dominios de entradas y salida en conjuntos de rangos de datos (particiones equivalentes)
- **Análisis del valor límite** está estrechamente ligado al particionamiento equivalente en el que selecciona datos desde los **límites** de la partición, se basa en que las fallas tienden a encontrarse en los extremos de los rangos.

Enfoque caja negra

- Desventajas
  - Desconocimiento de cobertura de código.
  - No proporciona información para localización de defectos.
  - Menor probabilidad de detección de fallas en el código.
- Ventajas
  - No consume demasiado tiempo.
  - Ponen énfasis en los requerimientos.
  - No requiere conocimiento de las implementaciones.
  - Diseño Temprano de Pruebas
  - Útil para pruebas de regresión.

Enfoque caja blanca

- las técnicas que derivan casos de prueba mediante el examen de la construcción del programa (su "trabajo interno") fueron llamadas técnicas de "caja blanca".
  - También denominadas como "técnicas estructurales" o "basadas en código o aún técnicas de "caja de cristal".

Enfoques caja blanca

- Las técnicas de caja blanca tienden a concentrarse en la estructura del programa y alientan al examinador a generar datos para ejecutar el programa con distintos grados de minuciosidad o cobertura:
  - decisiones
  - instrucciones - bloques de instrucciones
  - variables
  - estados

Enfoque de caja blanca

- Enfoque de caja blanca permite comprobar la ejecución de :
  - todos los caminos independientes de cada módulo
  - todas las decisiones lógicas en sus vertientes verdaderas y falsas
  - todos los ciclos en sus límites
  - todos los resultados o estados intermedios válidos
  - todas las estructuras de datos internas válidas

Técnicas enfoque Caja Blanca

- Basadas en el flujo de control del programa
  - Expresan los cubrimientos de las pruebas en términos del grafo de flujo de control del programa
- Basadas en el flujo de datos del programa
  - Expresan los cubrimientos de las pruebas en términos de las asociaciones definición-uso del programa
- Mutation testing
  - Se basan en crear mutaciones del programa original provocando distintos cambios en este último
    - Si el dato de prueba revela la falla introducida, entonces el mutante se dice que ha sido muerto y los datos de prueba son adecuados.
    - Sin embargo, si las fallas no son detectadas entonces los datos de prueba son inadecuados y tienen que ser aumentados para detectar otra mutación.

Complejidad Ciclomática

- La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa.
  - En el contexto de la prueba del camino básico, la complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

Complejidad Ciclomática

- Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición.
- En términos del grafo de flujo, un camino independiente está constituido por lo menos por una arista que no haya sido recorrida anteriormente a la definición del camino.

Complejidad Ciclomática

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
- La complejidad ciclomática, V(G), de un grafo de flujo G se define como:

  $V(G) = A - N + 2$

- Donde A es el número de aristas del grafo de flujo y N es el número de nodos.

Complejidad Ciclomática

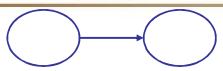
- La complejidad ciclomática, V(G), de un grafo de flujo G también se define como:

  $V(G) = P + 1$

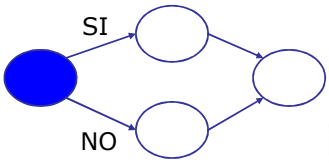
- Donde P es el número de nodos predicado contenidos en el grafo de flujo G.

Notación

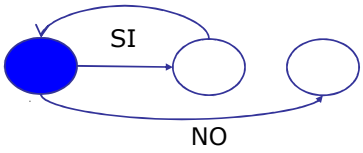
Secuencia



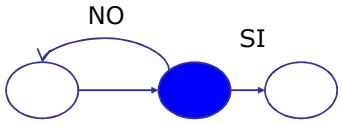
Condición IF



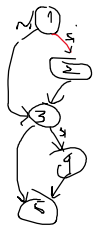
Bucle While



Bucle Hasta




```
If (a > 1)
  x = x / a
If (a = 2) or (b > 1)
   x = x + 1
```

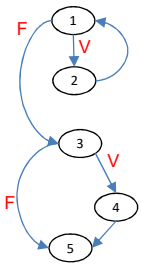


A-N+2
6-5+2
3

C1:1,3,4,5
C2:1,2,3,5
C3:1,2,3,4,5
C3:1,3,5
(Todas Comb 4) Mínimo 3

```
If (a > 1) and (b = 0)
    x = x / a
ElseIf (a = 2)
x = x + 1
```

```
For(i=1;n (i<=n);++) __1
    x = x / i _____2
If (a = 2) or (x > 1)__3
x = x + 1____4
}____5
```



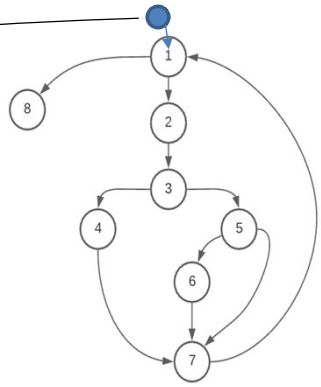
A-N+2
6-5+2
3

135
1345
12135

```
i=1
While(x>0)
{x = x / i
If (a > 1) and (b = 0)
    x = x / a
ElseIf (a = 2)
x = x + 1
i=i+1
}
```

i=1, x=a F(a,b)

```
While(x>0)
{x = x / i
    If (a > 1) and (b = 0)
    x = -x / a
Else
    If (a = 2)
            x = x - 1
i=i+1
}
```



a	b	x	i	X>0	A>0	and	B=0	A=2	
2	0	-1	2	v	v	v	v		01234718
-1	0	-1	1	f	f	f	v		018
No se puede probar- error código inalcanzable									
2	1				v	f	f		012356718
					f	f	f		

Caja Blanca: Criterio de cubrimiento de sentencias

- Asegura que el conjunto de casos de pruebas (CCP) ejecuta al menos una vez cada instrucción del código

```
If (a > 1) and (b = 0) {
    x = x / a
}
If (a = 2) or (x > 1) {
    x = x + 1
}
```

Este criterio es tan débil que normalmente se lo considera inútil. Es necesario pero no suficiente (Myers)

Datos de Prueba
a=2, b=0, x=3

Caja Blanca: Criterio de cubrimiento de decisión

– Cada decisión dentro del código toma al menos una vez el valor true y otra vez el valor false para el CCP

(a = 2) or (x > 1)

v f

```
If (a > 1) and (b = 0) {
    x = x / a
}
If (a = 2) or (x > 1) {
    x = x + 1
}
```

a > 1	b = 0	and

Es más fino que el criterio de sentencias

Datos de Prueba 1 a=3, b=0, x=3

Datos de Prueba 2 a=2, b=1, x=1

Caja Blanca: cubrimiento de condición

– Cada condición dentro de una decisión debe tomar al menos una vez el valor true y otra el false para el CCP

(a = 2) or (x > 1)

v f

```
If (a > 1) and (b = 0)
{x = x / a}
If (a = 2) or (x > 1)
 {x = x + 1}
```

a > 1	b = 0	

Hay que tener casos tal que a>1, a<=1, b=0 y b<>0 en el punto a y casos en los cuales a=2, a<>2, x>1 y x<=1 en el punto b

Este criterio es generalmente más fino que el de decisión

Datos de Prueba a=2, b=0, x=4

Datos de Prueba a=1, b=1, x=1

Caja Blanca: Criterio de cubrimiento de decisión / condición

- Combinación de los dos criterios anteriores

(a = 2) or (x > 1)

v f

If (a > 1) and (b = 0)
{x = x / a}
If (a = 2) or (x > 1)
{x = x + 1}

a > 1	b = 0	and

Es un criterio bastante fino dentro de caja blanca (Myers)

Datos de Prueba a=2, b=0, x=4

Datos de Prueba a=1, b=1, x=1

Caja Blanca: Criterio de cubrimiento de condición múltiple

- Todas las combinaciones posibles de resultados de condición dentro de una decisión se ejecuten al menos una vez

If (a > 1) and (b = 0)
{x = x / a}
If (a = 2) or (x > 1)
{x = x + 1}

a > 1	b = 0	and

a = 2	x > 1	or

Incluye al criterio de condición/decisión.

Myers lo considera un criterio aceptable

Satisfacer 8 combinaciones:

1) a>1, b=0 2) a>1, b<>0
3) a<=1, b=0 4) a<=1, b<>0
5) a=2, x>1 6) a=2, x<=1
7) a<>2, x>1 8) a<>2, x<=1

Los casos 5 a 8 aplican en el punto b

Datos de Prueba a=2, b=0, x=4 cubre 1 y 5

Datos de Prueba a=2, b=1, x=1 cubre 2 y 6

Datos de Prueba a=1, b=0, x=2 cubre 3 y 7

Datos de Prueba a=1, b=1, x=1 cubre 4 y 8

Caja Blanca: Criterio de cubrimiento de arcos

- Se “pasa” al menos una vez por cada arco del grafo de flujo de control del programa
- El criterio no especifica con qué grafo se trabaja; el común o el extendido (grafo de flujo de control dónde está dividida cada decisión en decisiones simples)
- Si es con el común este criterio es igual al de decisión

Caja Blanca: Criterio de cubrimiento de trayectorias independientes

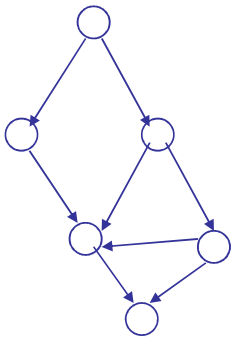
- Ejecutar al menos una vez cada trayectoria independiente

El número de trayectorias independientes se calcula usando la complejidad ciclomática

CC = Arcos – Nodos + 2

El CC da el número mínimo de casos de prueba necesarios para probar todas las trayectorias independientes

La cantidad de casos de prueba suele ser demasiado grande. Es un criterio de referencia



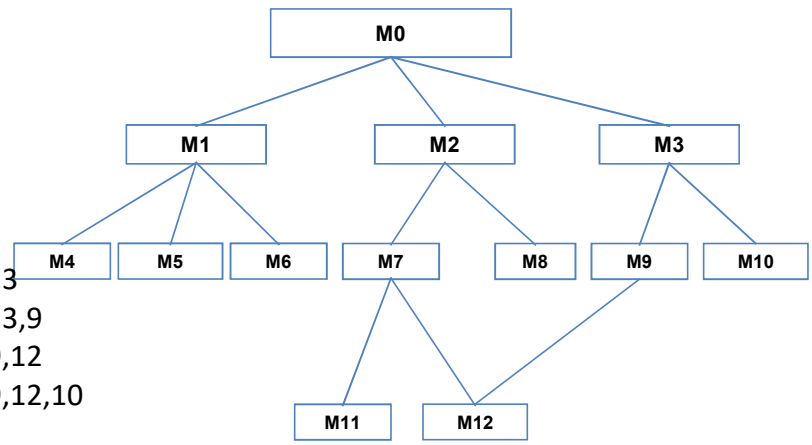
Si se divide en decisiones de una única condición (esto no está especificado en el criterio), es el más fino de los vistos hasta ahora.

Enfoque Caja Blanca

- Desventajas
  - Consume demasiado tiempo.
  - Pueden perderse de vista aspectos de la funcionalidad.
  - Requiere un entendimiento profundo de la implementación del software.
  - Algunas veces es difícil forzar ciertas condiciones de la estructura de los programas.
- Ventajas
  - Mayor probabilidad de detección de fallas.
  - Ayuda a localizar defectos.
  - Gran apoyo de herramientas automatizadas.

Arquitectura

Desc en profundidad
0 1
0,1,4
0,1,4,5,
0,1,4,5,6
0,1,4,5,6,2
0,1,4,5,6,2,7
0,1,4,5,6,2,7, 11
0,1,4,5,6,2,7,11, ~~12~~
0,1,4,5,6,2,7,11,~~12~~,8
0,1,4,5,6,2,7,11,~~12~~,8,3
0,1,4,5,6,2,7,11,~~12~~,8,3,9
0,1,4,5,6,2,7,11,8,3,9,12
0,1,4,5,6,2,7,11,8,3,9,12,10



Resguardos
4,5,6,
5,6
6
--
7,8
8,11,12
8,12

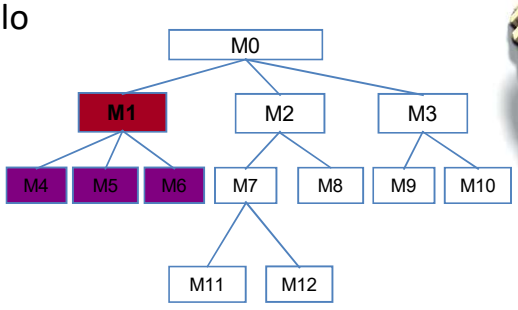
12
12,9,10
12,10
10

Proceso de Integración Descendente

- Se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal (programa principal).
- Los módulos subordinados se van incorporando en la estructura al módulo de control principal, ya sea *primero en profundidad* o bien *primero en anchura*.

Proceso de Integración Descendente

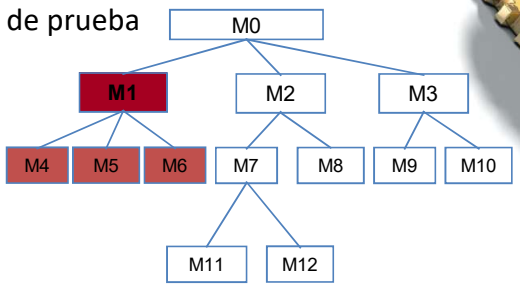
- Para probar un modulo se van disponiendo de resguardos para todos los módulos directamente subordinados al módulo



- se van sustituyendo los resguardos subordinados uno por uno por los módulos reales, los cuales se prueban

Proceso de Integración Descendente

- Después de la integración de un módulo, es necesario realizar pruebas de regresión:
  - repitiendo casos de pruebas
  - generando nuevos casos de prueba



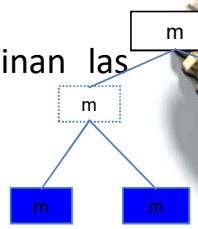
.. De manera tal de “reducir la siembra de nuevos errores”

Integración Ascendente

- La prueba de integración ascendente comienza la construcción y la prueba con los módulos atómicos, es decir con los módulos de nivel más bajo en el sistema.

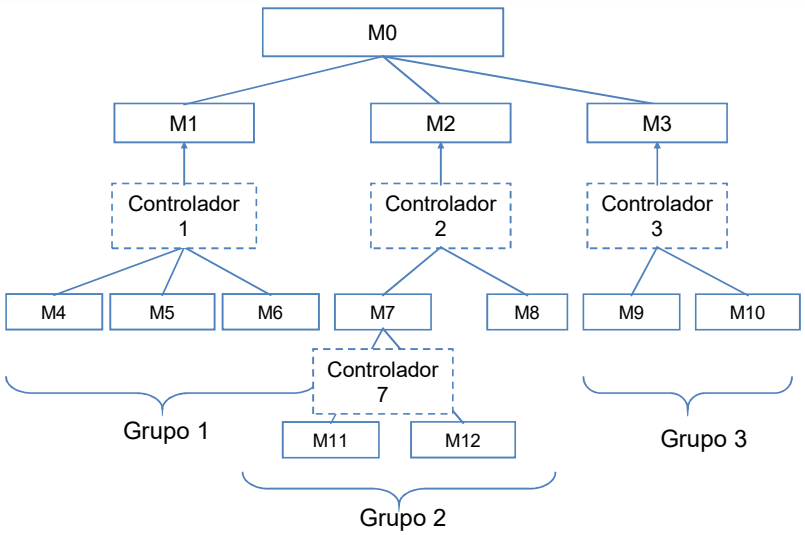
Proceso Integración Ascendente

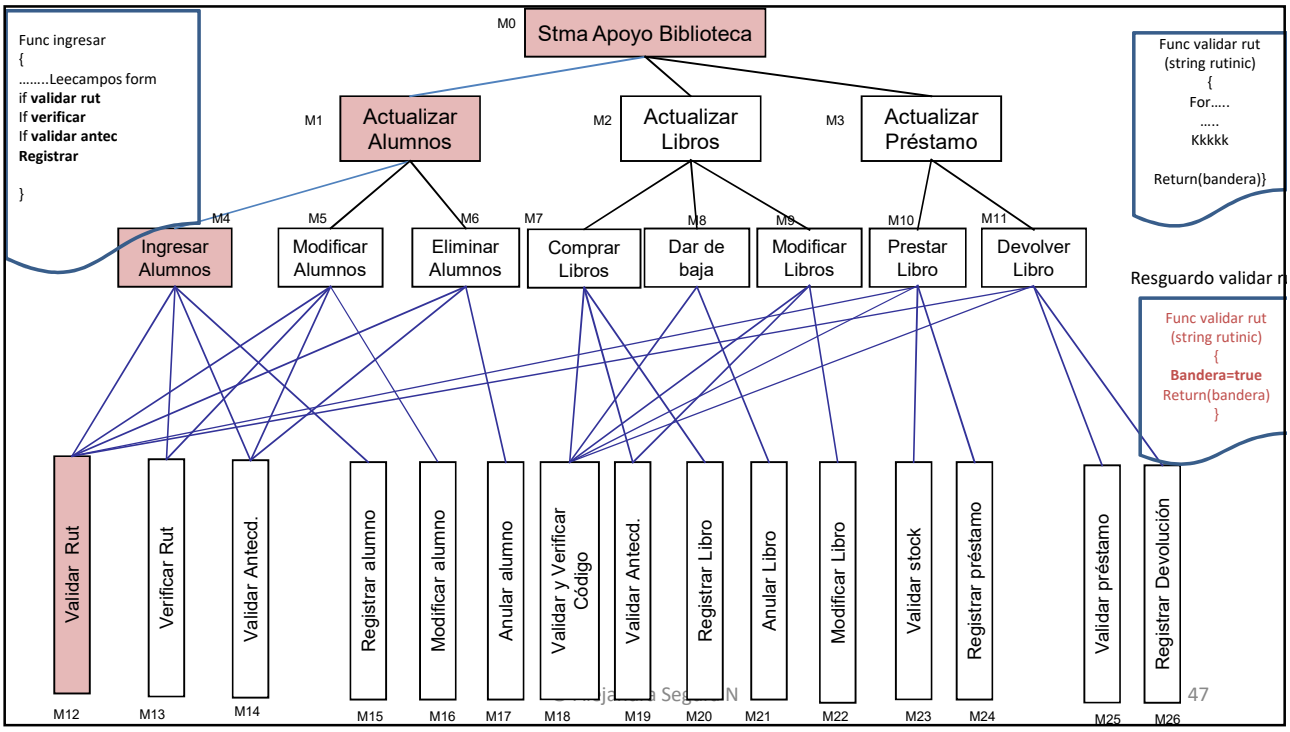
- Asociar los módulos que realicen una función específica de software,
- Se disponen de controladores que coordinan las relaciones entre ellos.

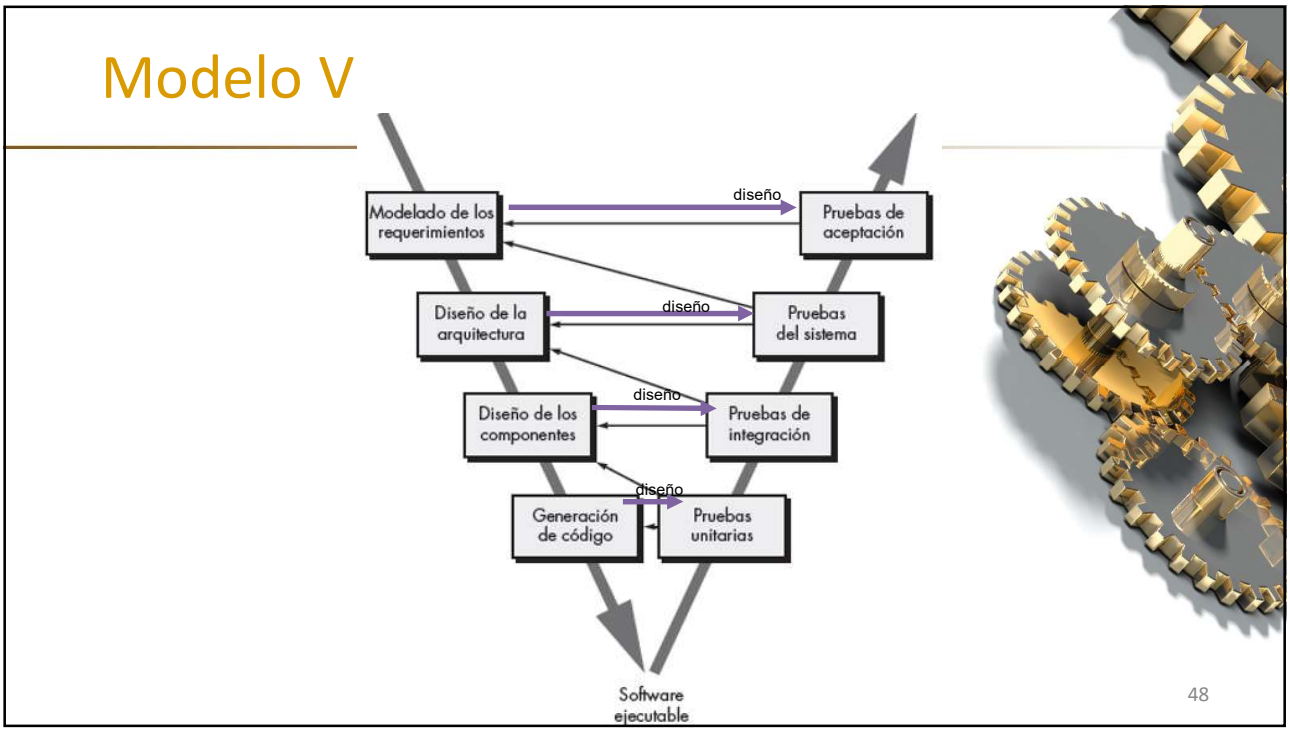


- Una vez probados, se eliminan los controladores y se continúa completando la integración

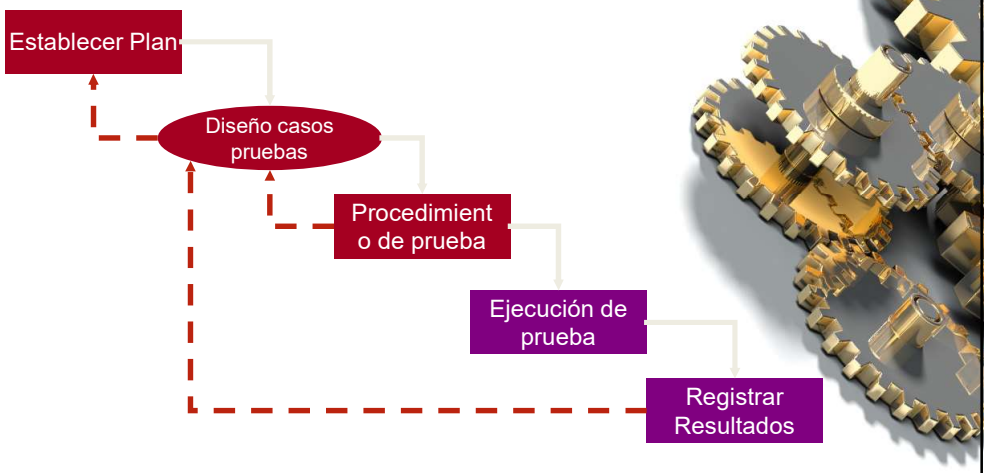
Integración Ascendente




Func ingresar
{
……..Leecampos form
if validar rut
If verificar
If validar antec
Registrar
}
M0
Stma Apoyo Biblioteca
M1
Actualizar Alumnos
M2
Actualizar Libros
M3
Actualizar Préstamo
Func validar rut (string rutinic)
{
For…..
…..
Kkkkk
Return(bandera)}
M4
Ingresar Alumnos
M5
Modificar Alumnos
M6
Eliminar Alumnos
M7
Comprar Libros
M8
Dar de baja
M9
Modificar Libros
M10
Prestar Libro
M11
Devolver Libro
Resguardo validar r
Func validar rut (string rutinic)
{
Bandera=true
Return(bandera)
}
Validar Rut
Verificar Rut
Validar Antecd.
Registrar alumno
Modificar alumno
Anular alumno
Validar y Verificar Código
Validar Antecd.
Registrar Libro
Anular Libro
Modificar Libro
Validar stock
Registrar préstamo
Validar préstamo
Registrar Devolución
M12
M13
M14
M15
M16
M17
M18
M19
M20
M21
M22
M23
M24
M25
M26
47


Modelo V
Modelado de los requerimientos
diseño
Pruebas de aceptación
Diseño de la arquitectura
diseño
Pruebas del sistema
Diseño de los componentes
diseño
Pruebas de integración
Generación de código
diseño
Pruebas unitarias
Software ejecutable
48

Proceso



Planificación de Prueba

- Planificación Global de la Prueba y General del Subproceso de Prueba: Actividad conducente a generar el plan de un subproceso de prueba específico. Por ejemplo, la planificación general de la prueba de unidad involucra la asignación de tiempos, recursos y responsables para la realización del subproceso de prueba de unidad.

Plan de Pruebas

- Plan de pruebas,
  - General
    - Para que se probará
    - Define que se probará,
    - Quienes lo harán,
    - Cuando se hará
  - Diseño de las pruebas
    - **Rasgos de prueba**
    - **Enfoque- Técnicas**
    - **Casos de prueba**
    - Configuración Requerida Software
    - Procedimiento de Prueba
  - Criterios de aceptación y fallos,

Diseño casos prueba unidad

Estructura propuesta para especificar casos prueba de unidad

RASGOS	UNIDAD	OBJETIVOS	CASO DE PRUEBA
Funcionalidad Calidad ... Interfaz		Requerimientos específicos Atributos de calidad	Restricciones De Configuración Hw, Sw , Redes Pre condiciones o requisitos Datos Entrada, Salida E Intermedios
Función de actualizar pago	**Módulo ingresa pago (mod_ing_og_v1)**	Ingresar pago por venta (R_FUN_01) Ingresar pago por compra (R_FUN_02)	•PC X; NAVEGADOR •DATOS de venta válidos, estado de venta en tramite •Caja negra Prueba funcional •Datos entrada •monto:75000, tipo v, código v_01 •Dato de salida •registro pago tabla PAGO (código, forma, detalle)

Procedimiento de prueba

- Especificación de Tareas desde :
  - Preparar configuración
  - Preparar medios de obtención de los datos de entrada
  - Ejecutar prueba
  - Registro de los resultados
  - Evaluación de resultados de acuerdo a criterios de aceptación

WALKTHROUGHS

- Walkthroughs estructurado, (Revisiones parciales del Software) (Yourdon, 1989)
- Los objetivos, filosofía y estados preparatorios, son ampliamente similares a los de la **técnica de inspección** aunque es **menos formal** y su **ejecución** es sutilmente **diferente**.

WALKTHROUGHS

- El walkthroughs involucra típicamente
  - un equipo de 3 a 5 personas
    - "autor" del programa quien guía el proceso,
    - "secretaria" para registrar los errores encontrados
    - "probador" para proporcionar los datos de prueba.
- "el autor" del programa guía a los otros en "ejecutar mentalmente" el código.

WALKTHROUGHS

- "probador " proporciona un par de documentos de casos de prueba –ejemplo de entradas y salidas esperadas – para estimular el análisis de la lógica del programa y los supuestos.
- El walkthrough entonces procede monitoreando el estado de las variables del programa, sobre el papel o sobre una pizarra, tal como el programa sería ejecutado

Inspecciones

- Esta técnica puede ser aplicada a alguna representación del sistema (requerimientos, diseño, datos de pruebas, etc.) con la meta de descubrir anomalías y defectos
- No requiere la ejecución de un sistema; por eso pueden ser usadas antes de la implementación

Planear la inspección

- Se deben considerar los siguientes aspectos:
  - Lugar donde se realizan las inspecciones
  - Planeación detallada de la inspección, Cubre aspectos como:
    - El tipo de Inspección
    - La cantidad que se va a probar
    - El tipo de medición

¿Cuánta Inspección es Necesaria?

- El conocimiento previo que se requiere para decidir la cantidad de inspección incluye:
  - Historia de la calidad de los elementos del producto y proveedores
  - Lo crítico de los elementos en el cumplimiento del sistema
  - Información sobre la capacidad del proceso
  - Datos sobre las variables y condiciones del proceso

Exactitud en la Inspección

- La exactitud en la inspección depende de:
  - Que tan completa sea la planeación de la inspección
  - La desviación y precisión de los instrumentos
  - El nivel de error humano

Exactitud en la Inspección

- Medición de la exactitud del inspector
- La exactitud de la inspección aumenta con la repetición de la misma
- La exactitud de la inspección disminuye cuando el producto es muy complejo

Beneficios Esperados

	Situación Actual	Situación Propuesta
Técnicas de Verificación	**NO**	**SI**
Eficacia Pruebas de SW	**50%**	**75%**
Errores en Requerimientos	**100**	**100**
Defectos Detectados: **- Requerimientos** **- Diseño** **- Pruebas**	 **0** **0** **50**	 **50** **0** **38**
Defectos Liberados	**50**	**12**
Costo de Corrección	**3.350 UT**	**804 UT**

(Base Consultora América XXI)

Análisis de Código

- En un estudio de Fagan:
  - Con Inspección se detectó el 67% de las faltas detectadas
  - Al usar Inspección de Código se tuvieron 38% menos fallas (durante los primeros 7 meses de operación) que usando recorridas o walkthrough
- Ackerman et. al.:
  - reportaron que 93% de todas las faltas en aplicación de negocios fueron detectadas a partir de inspecciones
- Jones:
  - reportó que inspecciones de código permitieron detectar 85% del total de faltas detectadas

Inspecciones

- Horas-Hombre requeridas por defecto:
  - Inspecciones Formales: 0.7 HH.
  - Prueba Convencional: 5 a 18 HH.
- Es más económico corregir los defectos antes de que llegue a la etapa de depuración por ejecución,
  - por ejemplo
    - Software del Transbordador - Sistema de Aviación Primario desarrollado por IBM Houston entre 1982 y 1985 lograron reducir la **tasa de defectos de 2.25 a 0.08 defectos/KSLOC**

Inspecciones

- Eficiencia (defectos detectados por hora): fuente Hewlett-Packard
  - Prueba Convencional: 0.21 dd x hh
  - Caja Negra: 0.282 0.21 dd x hh
  - Caja Blanca: 0.322 0.21 dd x hh
  - Inspecciones: 1.057 0.21 dd x hh

Inspecciones

- Guía para la selección de inspectores:
  - Pares que ejecutan labores idénticas a las del autor
  - Quienes van a recibir el producto para seguir elaborándolo
  - Colegas del Grupo de Prueba/QA "Quality Assurance"
  - Autores de etapas anteriores en el desarrollo

Inspecciones FAGAN, Etapas

- Etapa de Planificación:
  - El moderador debe coordinar con el autor el producto a revisar y elegir los inspectores.
  - Se entrega a los inspectores el anuncio (con el producto a evaluar) donde se identifica al equipo, el producto a revisar, la sala de reunión y fecha
  - Se comienza a llevar estadística de los tiempos empleados en las distintas actividades

Inspecciones FAGAN, Etapas

- Etapa de Reseña:
  - Se hace esta actividad cuando el producto a revisar esta en etapa de requisitos y se necesita que los inspectores estén mejor preparados.
  - Se hace cuando las inspecciones son nuevas al proyecto.
  - Se hace cuando el autor esta usando nuevas técnicas de programación
  - Dura a lo más cuarenta minutos.

Inspecciones FAGAN, Etapas

- Etapa de Preparación:
  - Cada inspector revisa el producto a conciencia
  - Anota los problemas potenciales en su log de defectos
  - Se dan de tres a cinco días hábiles para prepararse para la inspección
  - Cada inspector debe registrar el tiempo que empleo en la preparación
  - El log se entrega al Moderador al menos medio día antes de la reunión

Inspecciones FAGAN, Etapas

- Etapa Reunión:
  - El Moderador conduce la reunión asegurando que se cumplen las normas, El Lector lee el producto, Cada inspector hace sus observaciones
  - Se discuten las observaciones y el secretario registra los defectos
  - Se clasifican los defectos
  - Los problemas que no se pueden resolver en cinco minutos se dejan pendiente para una reunión llamada **Tercer Tiempo**

Inspecciones FAGAN, Etapas

- Etapa Reunión:
  - El Moderador registra los tiempos de preparación de los inspectores
  - Se entrega al autor la lista de defectos
  - Se planifica la reunión de seguimiento
- Etapa Corrección de Errores:
  - Todos los defectos graves deben ser corregidos por el autor
  - Defectos menores y triviales son corregidos por los mismos inspectores

Inspecciones FAGAN, Etapas

- Etapa Tercer Tiempo:
  - Se discuten en forma informal los problemas anotados.
  - Se determina si tienen injerencia sobre el producto.
  - Aquellos problemas que requieren cambiar el producto se clasifican como defectos

Inspecciones FAGAN, Etapas

- Etapa Seguimiento:
  - El Moderador con el autor revisan las correcciones.
  - Se revisa por si se ha introducido nuevos errores.
  - El Moderador hace un informe de las actividades según los formularios definidos
  - El producto se clasifica como certificado por Inspección Formal