Pruebas de Software

Prof. José Guerrero
DCC, Universidad de Chile
Octubre-Noviembre
2008

Contenido del Curso

1. Introducción
2. Estadísticas Relativas al Proceso de Pruebas
3. Estándares Relativos a la Prueba de Software
4. Técnicas de Pruebas
5. Software CAST
6. Una Metodología de Pruebas

Introducción

- Reflexiones preliminares sobre la prueba de software

- Una autoevaluación
  La prueba de software no es asunto trivial.

Reflexiones Preliminares

- Una de las actividades más importantes del desarrollo.
- No se desarrollan planes de pruebas.
- En la universidades no se enseña el tema.
- Inexistencia de cursos y literatura especializada.
- Mitos y leyendas
- Lo primero que se sacrifica son las pruebas.

EJERCICIO Nº1

Autoevaluación:
La prueba de software no es un asunto trivial.

- Un programa recibe 3 números enteros por pantalla, los que son interpretados como representaciones de las longitudes de los lados de un triángulo.

- El programa escribe un mensaje que informa si el triángulo es escaleno, isósceles o equilátero.

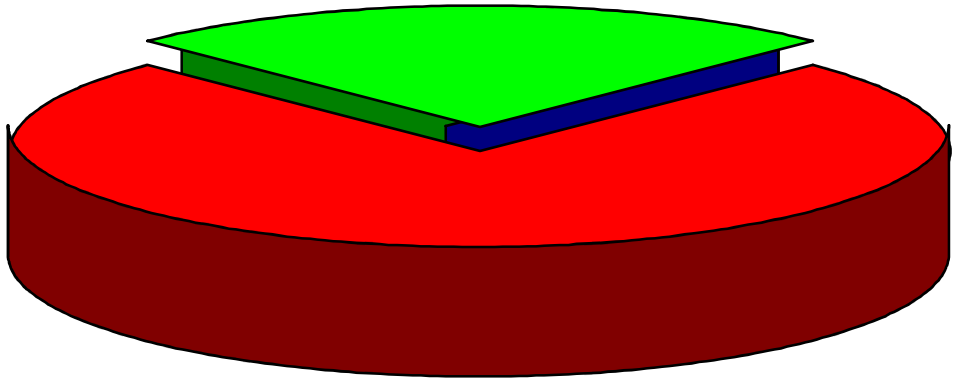
- Se solicita escribir los casos de prueba que Ud. estima necesarios para probar este programa.

Estadísticas Relativas a las Pruebas

Estadísticas relativas a las pruebas

Utilización de Recursos

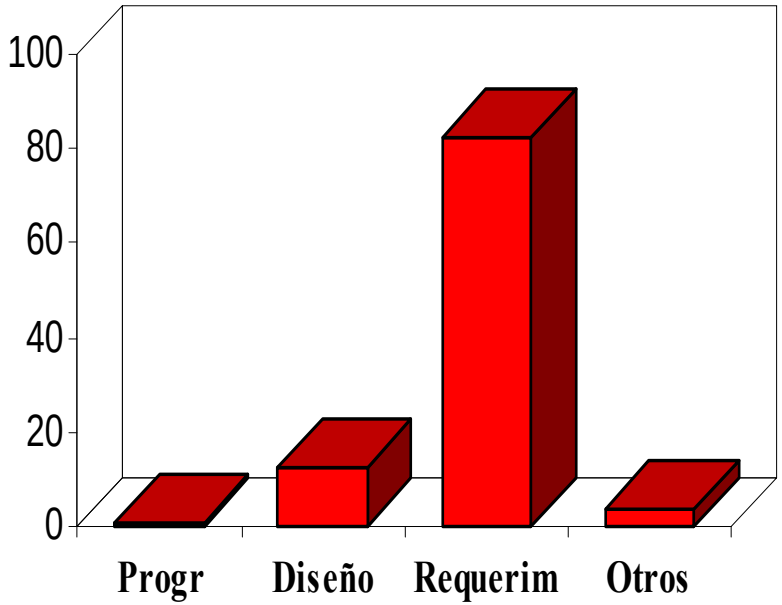
- El 75% de los recursos de informática se destinan a mantención de sistemas
- Sólo el 25% de los recursos se destinan al desarrollo de nuevas aplicaciones



Fuente: Software Engineering
Auerbach

Estadísticas relativas a las pruebas

Distribución Esfuerzo Mantención

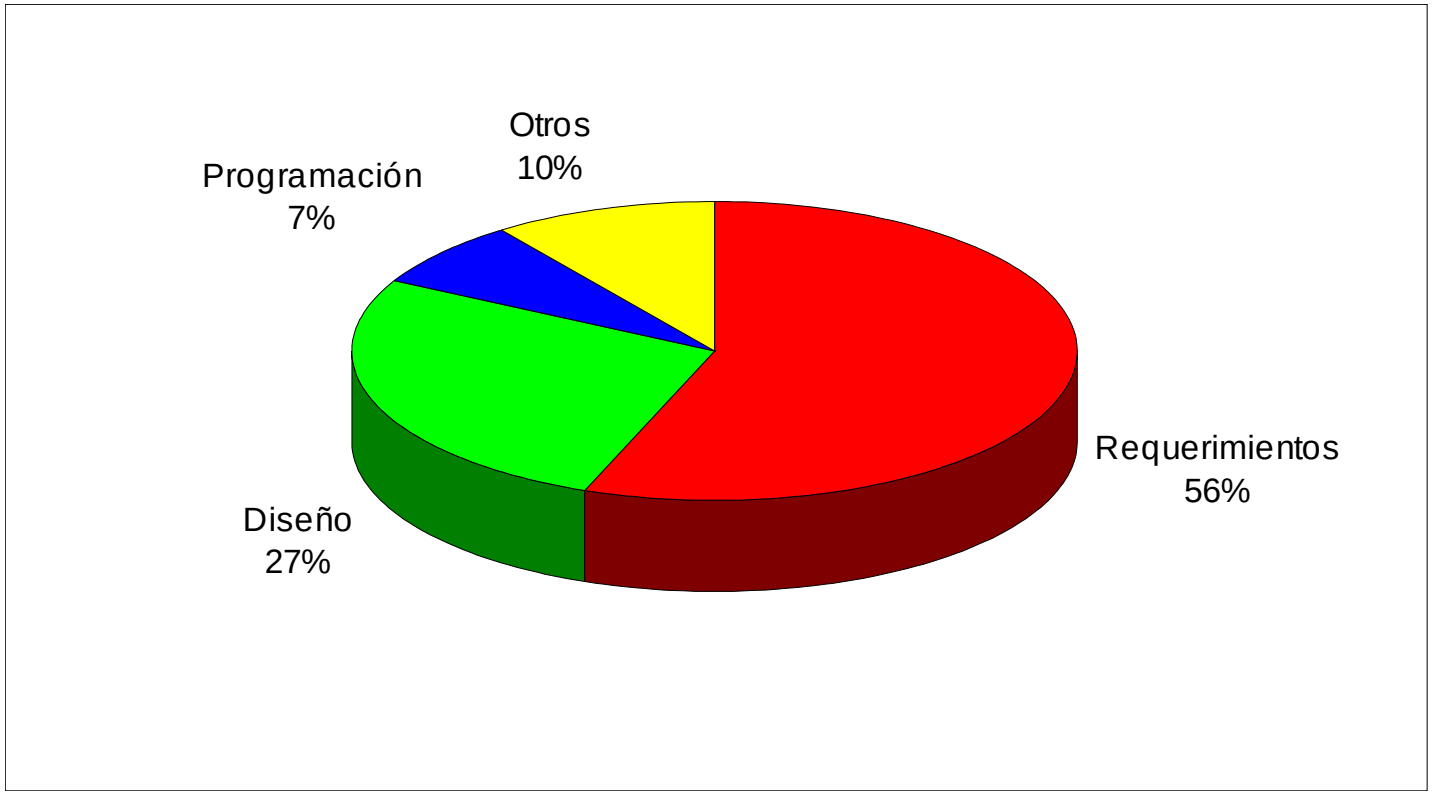


- Más del 80% de los recursos de mantención se destinan a corregir problemas que provienen de la Fase de Requerimientos

Fuente: An Introduction to Information Engineering
Clive Filkenstein

Estadísticas relativas a las pruebas

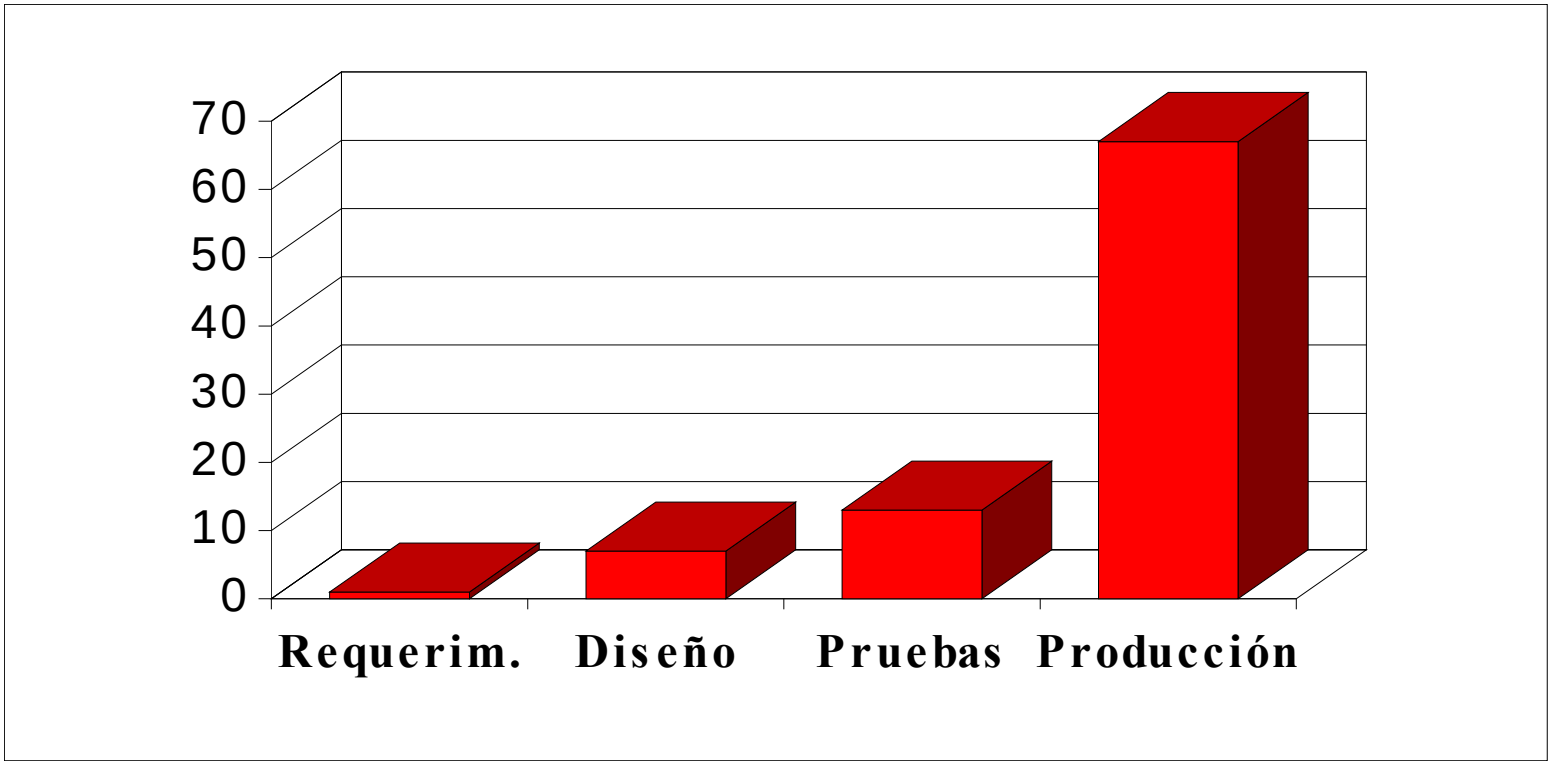
Fuentes de errores



Fuente: An Introduction to Information Engineering
Clive Filkenstein

Estadísticas relativas a las pruebas

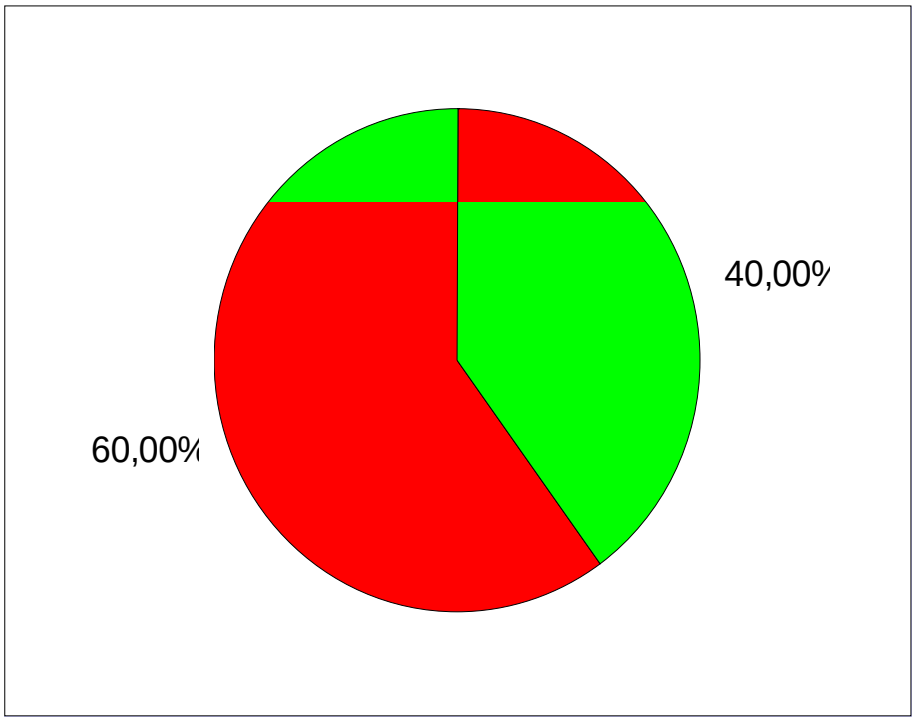
Costo de corrección de errores



Fuente: Ingeniería de Software
R. Pressman

Estadísticas relativas a las pruebas

Cobertura de Pruebas

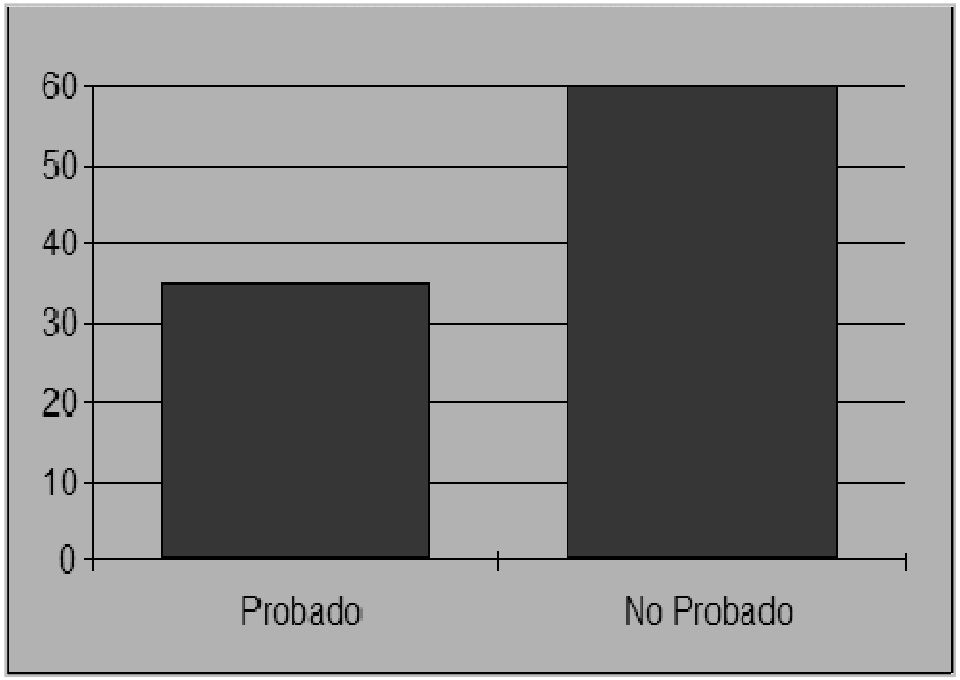


La cobertura de las técnicas tradicionales alcanza solamente entre un 40 a un 60% de las Variaciones Funcionales

Fuente: Bender & Associates, Inc. USA

Estadísticas relativas a las pruebas

Cobertura de Pruebas



- La cobertura de las técnicas tradicionales alcanza solamente entre un 20 a un 50% de las líneas de código

Fuente: Bender & Associates, Inc. USA

Estadísticas relativas a las pruebas

Código fuente por Punto de Función

Lenguaje	Valor más bajo	Promedio	Valor más alto
Ada83	60	71	80
Assembler(macro)	130	213	300
Visual Basic	63	98	135
C	60	128	170
C++	30	90	145
Cobol	65	107	160
Fortran	75	105	157
Pascal	50	91	125
Generadores	10	16	20
SmallTalk	12	21	28

Fuente: Software Productivity Research

Estadísticas relativas a las pruebas

Resultados del uso de técnicas de verificación

Origen defectos	Defectos potenciales	Eficiencia de Detección	Defectos producción
Requerimientos	1.00	77%	0.23
Diseño	1.25	85%	0.19
Codificación	1.75	95%	0.09
Documentac.	0.60	80%	0.12
Corr. errores	0.40	70%	0.12
Total	5.00	85%	0.75

Los valores estan expresados en defectos/puntos de función

Notese que si asumimos 1 PF = 100 Loc, tenemos que los defectos inyectados son 50/1000 Loc

Fuentes: American Programmer, Caper Jones, Glenford Meyers

Estadísticas relativas a las pruebas

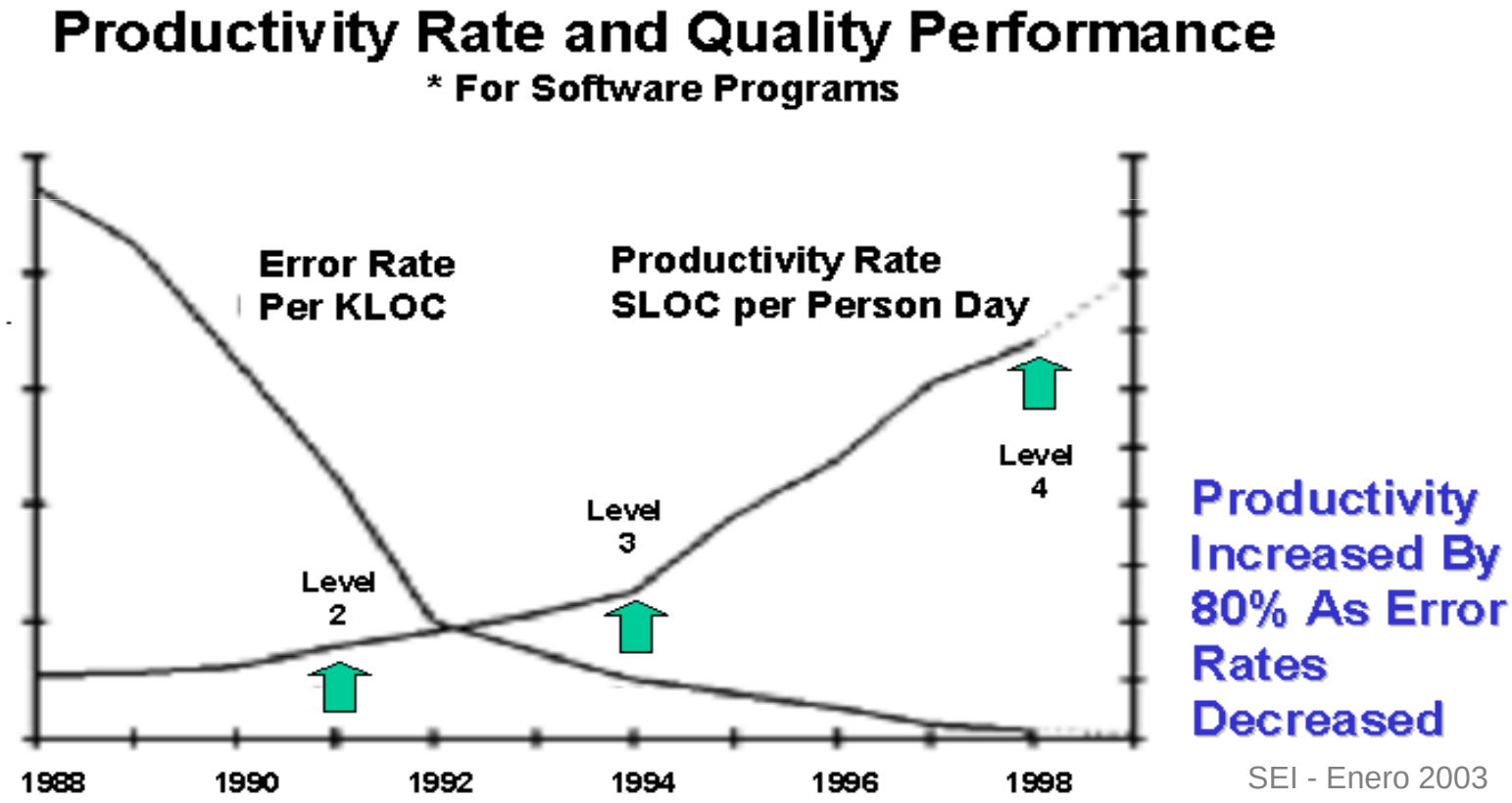
A que debemos aspirar

- El promedio de inyección de defectos es estimado en 50/1000 loc. (*)
- El objetivo del Testing es remover esos defectos.
- Las empresas como Motorola, Hitachi, HP y otras remueven sobre el 99% de los defectos.
- Estos valores pueden variar signigicativamente entre lenguajes, tamaño de sistemas, organizaciones, industrias, etc.

*Fuente: Glenford Meyers.

Estadísticas relativas a las pruebas

Aumento de Calidad y Productividad



SEI - Enero 2003

Estándares relativos a la prueba de software

Normas de Calidad ISO 9000

Normas de Calidad ISO 9000

- Son reglas básicas de calidad, independientes del producto o servicio de que se trate.

- Son un conjunto de buenas prácticas de fabricación de un producto u ofrecimiento de un servicio.

- Aseguran que el proveedor tiene la capacidad de producir los bienes o servicios requeridos.

Normas de Calidad ISO 9000

- En forma complementaria a las ISO 9000 existe el documento ISO 9000-3, el cual es una guía específica para la aplicación de la norma ISO 9001 al desarrollo y mantención de software.

- El ISO 9000-3 está dividido en 3 partes

  - Marco de trabajo
  - Actividades en el ciclo de vida
  - Actividades de apoyo

Conceptos de Pruebas ISO 9000

En la parte 2, y específicamente en lo relativo a las pruebas, la norma ISO 9000 establece los siguientes elementos principales:

a. Las pruebas deben ser realizadas en varios niveles, desde el ítem de software individual hasta el producto completo.

b. En algunas ocasiones la validación, la prueba de campos y la prueba de aceptación pueden ser una sola actividad.

c. El documento que describe el plan de pruebas puede ser un documento independiente o estar compuesto de varios documentos.

Normas de Calidad ISO 9000

Planificación de las pruebas

Se deben especificar los siguientes elementos :

a. Planes para realizar las pruebas de unitarias, integración, sistema y aceptación

b. Definición de casos de pruebas, datos de prueba y resultados esperados.

c. Tipos de pruebas a realizarse, por ejemplo, prueba funcional, prueba de límites, pruebas de rendimiento, pruebas de usabilidad.

d. Especificar ambiente de pruebas, herramientas y técnicas de pruebas de software.

e. Criterios de completitud en las pruebas.

f. Documentación disponible (reuqrimientos, diseño, otros)

g. Personal y entrenamiento requerido

Normas de Calidad ISO 9000

Relativo a las pruebas

Especial atención se debe poner a los siguientes aspectos de las pruebas:

a. Los resultados de las pruebas deben ser registrados. (Formularios de Registro).

b. Cualquier problema descubierto y sus posibles impactos en otras partes del software deben ser anotados y sus responsables notificados para que el problema pueda ser seguido hasta su corrección. (Seguimiento de errores)

c. Las áreas impactadas por cualquier modificación deben ser identificadas y vueltas a probar. (Criterios para Test de Regresión)

d. La adecuación e importancia de las pruebas debe ser evaluada.
(Análisis Transaccional)

e. La administración de la configuración de hardware y software debe ser considerada y documentada. (Metodología de SCM)

Normas de Calidad ISO 9000

Relativo a la validación del Proveedor

Antes de entregar el producto y la prueba de aceptación del usuario, el proveedor debe validar su operación como un producto completo y si es posible bajo condiciones similares al ambiente que tendrá la aplicación de acuerdo a lo especificado en el contrato.

Relativo a la aceptación del comprador

Cuando el proveedor está listo para entregar el producto validado, el comprador debe juzgar si el producto es aceptable de acuerdo a los criterios estipulados en el contrato.

El método de manejar los problemas detectados durante la prueba de aceptación debe ser acordado entre el comprador y el proveedor y documentado.

Relativo a la aceptación del comprador (cont.)

Planificación de la prueba de aceptación

Antes de realizar las actividades de aceptación, el proveedor debe asistir al comprador en la identificación de lo siguiente:

a. Planificación del tiempo

b. Recursos y medio ambiente de hardware y software necesario

c. Criterios de aceptación

Normas de Calidad ISO 9000

Recursos y personal para verificación

El proveedor deberá identificar internamente los requerimientos, proveer los recursos adecuados y asignar personal entrenado para las actividades de verificación.

Las actividades de verificación deben incluir inspección, PRUEBAS, y monitoreo del diseño, producción, instalación, y procesos de servicio o productos. Revisiones del diseño y auditorías del sistema de calidad, procesos y/o productos deben ser realizadas por personal independiente de aquel que tiene directa responsabilidad por el trabajo que está siendo desarrollado.

Modelo de Madurez
CMMI

Capability Maturity Model Integration

Modelo de Madurez del Proceso de Desarrollo

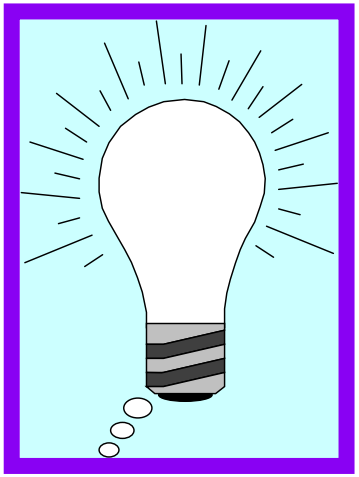
Origen del Modelo

- Departamento de Defensa, USA.
- Creación del Instituto de Ingeniería de Software (S.E.I)
- Metodología de evaluación de proveedores
- Desarrollo del Modelo CMM
- Desarrollo del Modelo CMMI

Importancia del Modelo

- Metodología de mejoramiento del Proceso de Desarrollo
- El mayor aporte para superar la “crisis del software”

¿ Cuál es la idea ?

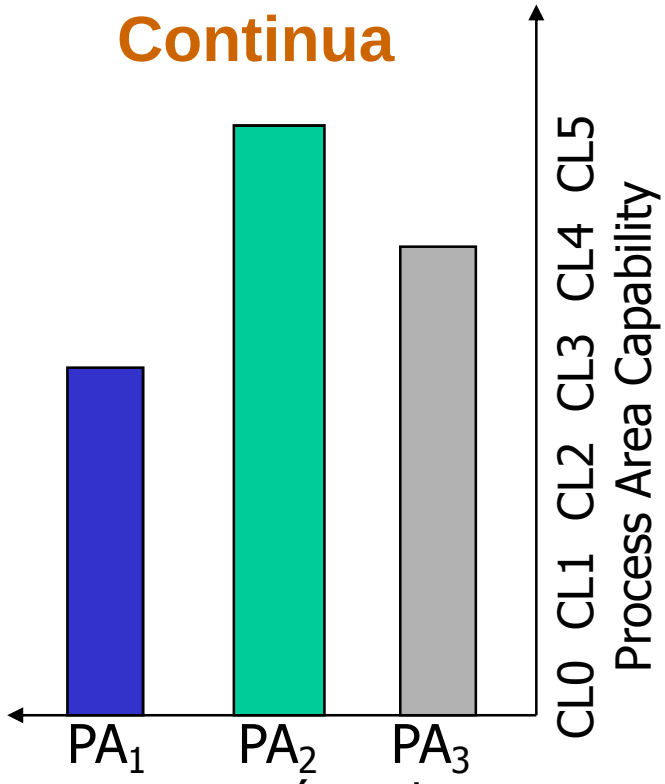


- **Evaluar el proceso**
  - ¿Cuán “bueno” es mi actual proceso ?
  - ¿Cuáles son mis fortalezas y debilidades?
  - ¿Qué debo hacer para mejorar ?

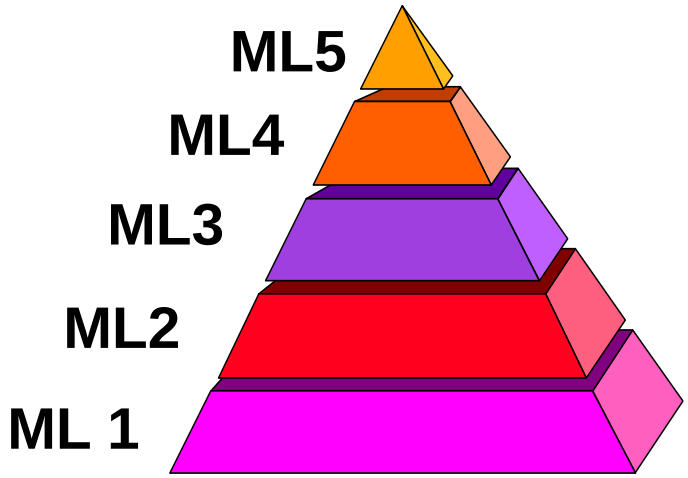
- **... y, mejorarlo**
  - Pasos requeridos para iniciar el cambio
  - Acciones que resultan más rentables
  - Métodos para un control cuantitativo
  - Como ir de un enfoque de corrección de problemas a un enfoque de prevención

Modelo de Madurez del Proceso de Desarrollo

Continua



...para un área de proceso
o un conjunto de áreas de proceso



...para un conjunto de áreas de proceso
a través de la organización

Los 5 Estados del modelo CMMI

Inicial (Nivel 1)

Este estado se caracteriza porque la organización opera sin procedimientos formales, sin estimación de costos ni planes de proyectos. Las herramientas no están bien integradas ni uniformemente aplicadas. El control de cambios no es riguroso. La administración no cuenta con buena información acerca de los problemas y asuntos internos, dado que muchos problemas son diferidos y aun olvidados, la instalación y mantención de software a menudo presenta problemas. Aun cuando muchas organizaciones tienen procedimientos formales para planificar y monitorear su trabajo, no disponen de un mecanismo que les asegure que ellos son usados.

Administrado (Nivel 2)

En este estado la organización ha logrado un rendimiento estable a través de una administración de compromisos, costos, planificación y control de cambios. Esta estabilidad se traduce en un conjunto de acciones diseñadas para lograr controlar los costos y plazos. Dichas acciones involucran los siguientes aspectos:

- organización
- administración de proyectos
- administración del proceso de desarrollo
- tecnología.

Los 5 Estados del modelo CMMI

Definido (Nivel 3)

La organización ha definido un proceso estándar (metodología), el que es adecuado a las necesidades de cada proyecto. Este provee una base informada para manejar las contingencias, permitir mejorar ordenadamente y facilitar la introducción de tecnología de apoyo. La organización ha instituído un conjunto de métodos y prácticas que le permiten anticiparse a las contingencias. Existen escasos datos cuantitativos.

Administrado cuantitativamente (Nivel 4)

La organización ha desarrollado un completo proceso de análisis y medidas, tales como calidad, costo, productividad, etc. El preciso conocimiento de los procesos provee las bases para una significativa y continua calidad de sus productos y un mejoramiento de la productividad. Existe asignación de recursos para la recopilación y mantención de los datos.

Optimizado (Nivel 5)

En este estado, la completa información acerca del proceso de software es usada para evaluar la efectividad de este proceso y hacer ajustes en forma regular. Esto provee las bases para una contínua mejora del desarrollo y un ordenado y planificado aumento de la productividad.

Las Pruebas en el modelo CMMI

Entrenamiento

Se debe identificar las necesidades de capacitación que son necesarias para que los profesionales puedan realizar funciones relacionadas con las pruebas.

Aseguramiento de Calidad de Software(SQA)

Esta función es realizada por una o más personas y su misión es asegurar que todos los proyectos cumplen con los estándares definidos formalmente por la organización, en particular, para las actividades de prueba.

Ingeniería de Proceso

Esta función es realizada por una o más personas y su misión es definir los estándares, normas, técnicas y procedimientos del proceso de desarrollo que usará la organización. En particular, deben definir los estándares de la organizacion relacioandos con el proceso de pruebas.

Las Pruebas en el modelo CMMI

Registro de datos

Se deben realizar actividades de recopilación de los datos que permitan evaluar los resultados del proceso de pruebas, de manera de realizar un mejoramiento continuo de la calidad del mismo. La idea es utilizar la fórmula PDCA (Plan, Do, Check, Act). Esto se traduce en :

- Planificar lo que hay que hacer
- Hacer el trabajo requerido para lograrlo
- Chequear los resultados
- Actuar para mejorar basándose en lo aprendido.

Control estadístico del proceso

De la actividad de recopilación y análisis de datos se debe llegar a establecer un control estadístico del proceso. Esto quiere decir que la organización debe identificar el rango de variabilidad con respecto a un valor esperado. Por ejemplo, puede usarse el valor de la Desviación Estándar con respecto al valor medio.

Estándares IEEE

Estándares ANSI/IEEE

La aplicación de los conceptos definidos en las dos Normas siguientes ayudan a:

- Detectar y corregir errores temprano
- Reducir los costos y riesgos
- Mejorar la calidad y confiabilidad
- Mejorar la visión administrativa del proceso de desarrollo
- Cambios propuestos y sus consecuencias pueden ser evaluados rápidamente

Nota:

Las normas se han modificado en relación al texto original, incorporando la experiencia de su uso en la práctica a objeto de hacerlas aplicables rápidamente en las empresas nacionales.

Planes de Verificación y Validación

- Apoya en la tarea de definir una metodología de control de calidad de los productos de software a lo largo de todo el ciclo de vida.
- Para cada fase del ciclo de vida define entradas, tareas y salidas.
- Provee un estándar de contenido para cada verificación y validación

Estándar ANSI/IEEE 1012

Planes de Verificación y Validación

- Por ejemplo, para la fase Requerimientos:

Entradas

Estándares de documentación (IEEE, UML, Prototipos,...)
Especificaciones de Requerimientos
Documentación adicional provista por el usuario

Tareas o Actividades

Análisis de trazabilidad
Evaluación de los requerimientos (Correctitud, completitud,...)
Verificar el Plan de Pruebas (Si corresponde)

Salidas

Informe de cada tarea realizada (responsable, tiempos, % cumplimiento,...)
Informe de anomalías

Estándar ANSI/IEEE 1008

Pruebas de Unidad de Software

- Este estándar usado en combinación con el anterior, permite refinar una metodología de control de calidad, agregando conceptos detallados al interior de las fases definidas en la norma anterior.

- Si bien la Norma está orientada específicamente a las actividades de prueba unitaria de software, se recomienda su extensión a todas las fases del control de calidad durante el ciclo de vida del proyecto.

Estándar ANSI/IEEE 1008

Pruebas de Unidad de Software

- **Planificación**
  - Planificar estrategia, recursos y actividades
  - Determinar características a probar
  - Refinar el plan
- **Construcción de los casos de prueba**
  - Diseñar los casos de prueba
  - Transformar e implementar los casos en datos
- **Ejecución y medición del test**
  - Ejecutar las pruebas
  - Evaluar los resultados
- **Verificar el término**
  - Verificar término de tareas planificadas
  - Hacer informes
  - Despachar resultados

El proceso de pruebas

Lo que Ud. debe saber antes de comenzar a probar

Principios de Prueba

Definición

Prueba es el proceso de ejecutar un programa con el fin de encontrar errores

Esta definición implica que debe partirse de la suposición que el programa a probar contiene errores (lo cual casi siempre es válido) y se trata de encontrar tantos como sea posible.

Por el contrario, si nuestro objetivo es mostrar que el programa no con tiene errores, estaremos inconcientemente orientados hacia ese fin.

El adoptar una u otra actitud tiene profundas implicancias en el éxito de la prueba.

Principios de Prueba

Los principios de prueba a menudo parecen obvios y son ignorados, con serias consecuencias.

1. La definición del resultado esperado a la salida del programa es una parte integrante y necesaria de un caso de prueba.
   Muchos errores se originan por ignorar este principio. Si el resultado esperado de un caso de prueba no ha sido predefinido, existe la posibilidad de que un resultado plausible pero erróneo, se interprete como correcto.

2. Un programador debe evitar probar su propio programa
   La detección de errores es un proceso “destructivo”. Es por ello que es difícil para una persona adoptar la actitud mental necesaria para demostrar que lo que acaba de hacer está erróneo.
   Por otra parte, está el problema de falsas interpretaciones que puede haber hecho el programador. Su prueba no hará más que tratar de verificar lo correcto de su interpretación.

Principios de Prueba

3. Inspeccionar concienzudamente el resultado de cada prueba
   Este es probablemente el principio más obvio, pero, sin embargo, a menudo dejado de lado. empíricamente se ha encontrado que muchos programadores fracasan en la detección de errores, aún cuando en los resultados de las pruebas eran claramente observables.

4. Examinar un programa para comprobar que no hace lo que se supone que debe hacer es sólo la mitad del problema. La otra mitad consiste en ver si el programa hace lo que no se supone que debe hacer.
   Esto significa que los programas deben ser revisados con respecto a efectos colaterales no deseados.

5. Evitar los casos de prueba desechables a menos que el programa sea verdaderamente un programa desechable.
   Una práctica muy frecuente es sentarse frente al terminal e inventar casos de prueba “al ojo” .Cada vez que se prueba nuevamente, los casos de prueba deben ser reinventados. Generalmente no se hace, y las pruebas siguientes no son tan rigurosas como la primera vez. Esto significa que si se introdujo un error en una parte ya probada, rara vez se detecta.

Principios de Prueba

6. No planear el esfuerzo de prueba con la suposición tácita de que no se encontrarán errores.
   Este error se comete cuando se parte de la suposición de que la prueba es el proceso de mostrar que el programa funciona correctamente.

7. La probabilidad de encontrar errores adicionales en una sección del programa es proporcional al número de errores encontrados.
   Este fenómeno, tiene su base en comprobaciones empíricas. En efecto, los módulos de un programa (igualmente probados) que han presentado más errores tienen una probabilidad más alta de presentar otros errores.

8. La prueba de programas no es una ciencia exacta.
   Si bien existen métodos que ayudan en el proceso de elaboración de casos de prueba, ello requiere de todas maneras de una considerable cuota de creatividad.

Principios de Prueba

Importante

Las pruebas deben servir, en buena medida, para aprender de los errores cometidos.

Este enfoque significa que la organización debe preocuparse de crear las estructuras
necesarias para realizar una retroalimentación sistemática de las experiencias obtenidas en cada proyecto. Esto significa que deben investigarse, registrarse, difundirse y prevenirse las causas que originaron los defectos encontrados en las pruebas.

Esto conduce a mejorar el proceso de desarrollo, llevándolo de un esquema de corrección de errores a un enfoque de prevención de errores, generándose una disminución de las pruebas y un aumento de la productividad.

Técnicas de

Verificación y Validación

Definiciones

1. Verificación

Es el proceso de determinar si los productos de una fase dada del ciclo de desarrollo de software satisfacen los requerimientos establecidos durante la fase previa.

2. Validación

Es el proceso de evaluación de software al final del proceso de desarrollo para asegurar la concordancia del producto con los requerimientos especificados.

Técnicas de Verificación

Inspecciones

Es una de las técnicas más importantes en la detección de defectos en el Software. Se aplican, principalmente, a los productos de la fase de requerimientos, diseño, programación y plan de pruebas.

La idea es realizar una revisión en conjunto con el autor del producto a inspeccionar.

Objetivos

- Verificar que los productos cumplan con las especificaciones
- Detectar defectos
- Revisar alternativas
- Servir como entrenamiento

La técnica de inspección de código juega un rol principal en la prevención de defectos.

Caper Jones

Ejercicio Nº 2

Ud. debe hacer un viaje de 800 km en su vehículo. Haga una lista de chequeo para que su vehículo sea revisado y Ud. pueda estar tranquilo que no tendrá problemas.

Técnicas de Verificación

Cómo es el Proceso de Inspección ?

PROCESO DE INSPECCION	
Propósito	Identificación de defectos en productos de software
Entradas	Especificación del producto Estándares de productos de software Resumen del producto de software Tipos de defectos Checklists Listado de defectos de inspecciones anteriores
Actividad	Planificación Reunión inicial Preparación Reunión de inspección Corrección Seguimiento
Salidas	Productos de software corregidos Listado de defectos actualizados Estadísticas de proceso actualizadas

Técnicas de Verificación

ACTIVIDAD DE PLANIFICACION	
Propósito	Preparar todos los aspectos de la revisión
Entradas	Especificación del producto Estándares de productos de software Producto de Software Resumen del producto de software Tipos de defectos Checklists Listado de defectos de inspecciones anteriores
Actividad	Autor envía producto a revisión Moderador/autor revisan/coordinan aspectos globales Moderador selecciona inspectores Moderador planifica fechas Moderador coordina logística(salas, proyectores,..) Moderador distribuye productos de software Moderador distribuye estándares en caso necesario
Salidas	Checklist a aplicar Estándares a aplicar Notas para la revisión

Técnicas de Verificación

ACTIVIDAD REUNION INICIAL	
Propósito	Describir/Conocer el producto de software
Entradas	Producto de Software
Actividad	Autor realiza una descripción global del producto Moderador guía la reunión Autor distribuye material de apoyo Inspectores realizan preguntas Autor entrega/compromete información requerida
Salidas	Producto de software conocido

Técnicas de Verificación

ACTIVIDAD DE PREPARACION	
Propósito	Preparar los elementos a usar en la revisión
Entradas	Especificación del producto de software Estándares de productos de software Resumen del producto de software Tipos de defectos Checklists Listado de defectos de inspecciones anteriores
Actividad	Inspectores revisan los tipos de defectos Inspectores revisan los checklist Inspectores revisan los estándares Inspectores revisan las especificaciones Inspectores revisan los productos a inspeccionar Inspectores toman notas y preparan preguntas
Salidas	Checklist a aplicar Estándares a aplicar Notas para la revisión

Técnicas de Verificación

ACTIVIDAD DE INSPECCION	
Propósito	Identificar defectos en el producto de software
Entradas	Especificación del producto Estándares de productos de software Producto de Software Resumen del producto de software Tipos de defectos Checklists Listado de defectos de inspecciones anteriores
Actividad	El moderador guía la actividad El autor explica el producto paso a paso Inspectores reaizan preguntas Autor responde las preguntas Inspectores/Autor identifican defectos Inspector designado registra defectos Moderador revisa aspectos faltantes
Salidas	Informe de Defectos

Técnicas de Verificación

ACTIVIDAD DE CORRECCION	
Propósito	Corregir los defectos encontrados
Entradas	Listado de defectos Producto de software Estándares
Actividad	El autor revisa la lista de defectos El autor corrige los defectos El autor envía producto corregido al moderador
Salidas	Productos de software corregidos

Técnicas de Verificación

ACTIVIDAD DE SEGUIMIENTO	
Propósito	Verificar Corrección de defectos
Entradas	Listado de defectos Producto de software
Actividad	Moderador recibe producto corregido Moderador determina necesidad de una nueva inspección Moderador genera reportes de inspección Moderador actualiza estadísticas de proceso
Salidas	Productos de software corregidos Listado de defctos actualizados Estadísticas de proceso actualizadas

Recomendaciones sobre las Técnicas de Revisión e Inspección

A continuación se entregan recomendaciones que se aplican por igual a las técnicas de Revisión e Inspección

Estas recomendaciones son producto de la experiencia de aplicaciones en la práctica de cada una de estas técnicas.

Consideraciones a tener en cuenta en la realización

a. Se recomienda realizarla después de tener un producto “terminado” (Modelo de datos, DFD, carta estructurada, compilación sin errores).

b. Si se encuentran demasiados errores, suspender la inspección.

d. Los errores se deben registrar y no deben ser corregidos durante la sesión.

e. El uso que los jefes hagan de los resultados puede desvirtuar el propósito.

f. El autor y los participantes reciben aportes referentes a su estilo de diseño, programación, estándares de documentación, etc.

g. Cada sesión debe durar no más de dos horas

h. La tasa de revisión en inspección de código está entre 100 y 300 sentencias por hora. (Watts Humphrey)

Técnicas de Verificación

Ventajas

a. Tener un conocimiento cuantitativo de la calidad del proyecto

b. Criterios de término conocidos para cada fase del proyecto

c. Métodos para detectar errores, registrarlos y seguirlos hasta su corrección

d. Facilita y mejora sustancialmente la administración del proyecto

e. Los errores son detectados muy cercano al momento en que ellos se introducen

f. Contribuyen eficazmente a la recopilación de métricas

g. Mejoran la calidad del producto

h. Disminuyen los costos del proyecto

i. Permite avanzar desde un esquema de corrección de errores a un esquema de prevención de errores

Perfil del Moderador

- Contar con el entrenamiento requerido en inspecciones
- Imparcial con el trabajo, área y “persona” inspeccionada
- Tener experiencia en general en los temas bajo inspección
- No necesita ser experto en el tema específico inspeccionado
- Tener capacidad de dirección de reuniones
    - Mantiene la reunión en los temas que interesan
    - Evita posibles conflictos entre los participantes
    - Controla horarios
- Después de la actividad hace seguimiento de las observaciones hasta su corrección

Inspección de Requerimientos

Inspección de requerimientos

Objetivo específicos

- ⇨ Verificar que el producto final cumpla con los requerimientos especificados.
- ⇨ Verificar que los productos cumplan con criterios de correctitud, consistencia, etc.
- ⇨ Satisfacción de estándares, prácticas y convenciones.
- ⇨ Establecer la base para la próxima fase

Inspectores

- Moderador
- Analista que realizó el análisis
- Analista senior de otro proyecto
- Un programador
- Un representante de mantención
- Otro ?

Inspección de requerimientos

Entradas

- Estándares de documentación de la organización
- Especificación conceptual del sistema
- Especificación de requerimientos del sistema
- Documentos de interfaces del sistema
- Plan de calidad

Enfoque de la revisión

- Revisión de la trazabilidad de los requerimientos
- Evaluación de los requerimientos(Correctitud, consistencia, completitud, ...)
- Revisión de interfaces del sistema(Usabilidad, estándares, datos, ...)
- Revisión de los requerimientos versus el Plan de Calidad

Problemas del lenguaje natural

- Else ambiguo (falta el "Else")
  Ej: El código debe ser A, B o C.

- Ambigüedad de referencia
  Ej: Sume A a B. El número debe ser positivo.

- Omisiones de causas y efectos
  Ej: -Código 1 a 4 produce el mensaje. El Código también puede ser 5.
  - Ante algunas situaciones se debe reinicializar el campo en 0.
  - Si Ud. se pasa una luz roja entonces tendrá un parte.
  - Si la cuenta está sobregirada entonces rechazar el cheque.

- Operadores lógicos ambiguos
  Ej: Si el empleado es Profesional o tiene Asignación de Responsabilidad y tiene Asignación de Zona entonces tiene derecho a plan de salud tipo A.

Problemas del lenguaje natural

- Negaciones (ámbito, dobles)
  Ej: Si María no aprueba el examen, ella no pasará el curso.
  Si María aprueba el examen, ella pasará el curso.
  Nota: ¿Las dos frases significan lo mismo ?

- Frases ambiguas(verbos, adv, adj)
  Ej: La respuestas en pantalla deberán ser rápidas y oportunas.

- Variables ambiguas (nombres, alias)
  Ej: Los sueldos mayores que $100.000 tienen 2% de descuento. La remuneración resultante es imponible.

- Organización random (Causas y efectos mezclados)
  Ej: Si A o B entonces C. D es creado si E y F son enteros.

Inspección de requerimientos

Lista de palabras de potenciales ambigüedades

Condiciones	Ambigüedad de Referencia	Adjetivos Ambiguos	
Puede	Encima	Todos	Infrecuente
Podría	Debajo	Cualquier	Intuitivo
Es uno de	Tal	Apropiado	Inválido
Debe	El anterior	Costumbre	Muchos
Aconsejo	Estos	Eficiente	mayoría
Aconsejaría	Ellos	Pocos	Normal
Deberá	Este	Frecuente	Común
Debería	Esos	Mejorado	Raro
		Infrecuente	Mismo
		Intuitivo	Varios
		Varios	Parecido
		Parecido	Algunos
		Transparente	Estándar
		Típico	Usual
			Válido
Considerar en todos los casos los condicionales, *mente, género y número.			

Publicado con autorización de
Bender & Associates.
Queensbery
Nueva York. USA.

Inspección de requerimientos

Lista de palabras de potenciales ambigüedades

Adverbios Ambiguos		Variables Ambiguas	Ambigüedad Temporal
En consecuencia	Normalmente	La aplicación	Después
Casi	No absolutamente	El componente	Anualmente
Aproximadamente	A menudo	El dato	En un tiempo
Generalmente	Raras ocasiones	La base de	En el tiempo
Comúnmente	Raramente	El campo	Bimensual
Acostumbradamente	Aproximadamente	El archivo	Bisemanal
Eficientemente	raramente	El marco	Diariamente
Frecuentemente	Similarmente	La información	Mes por medio
Generalmente	A veces	El mensaje	Semana por
Casi nunca	Algo	El módulo	Rápido (a)
En general	Transparentemente	La página	En un rato
Infrecuentemente	Típicamente	La regla	Más tarde
Intuitivamente	Usualmente	La pantalla	Mensualmente
A menudo	Virtualmente	El estado	trimestralmente
Más o menos	Casi	EL sistema	Rápidamente
Sobre todo		La tabla	Pronto
		El valor	Dos veces al
		La ventana	Dos veces al
			Semanalmente
			Anualmente

Inspección de requerimientos

Lista de palabras de potenciales ambigüedades

Verbos Ambiguos		Casos Implícitos	Límites Ambiguos
Ajustar	Poder	También	Desde
Alterar	Minimizar	Aunque	Hasta
Enmendar	Poder	También	Hacia
Calcular	Modificar	Además	
Cambiar	Optimizar	Pero	**Amb. de**
Comparar	Ejecutar	Aunque	Etc.
Computar	Procesar	Para todos los	Etcetera
Convertir	Producir	Además	.....
Crear	Proveer	Sin embargo	
Adaptar	Soportar,Apoyar	Además de	
Derivar	Actualizar	Asimismo	
Determinar	Validar	Por otra parte	
Editar	Verificar	A pesar de	
Habilitar	Es decir, o sea	Por otra parte	
Facilitar	Rechazar	En caso contrario	
Mejorar		Aún	
Indicar		Aunque	
Manipular		A menos que	
Igualar, Emparejar		Mientras que	
Maximizar		Todavía	

Publicado con autorización de Bender & Associates. Queensbery Nueva York. USA.

Inspección de requerimientos

Ejercicio Nº 3

Inspeccione las siguientes reglas de proceso

La cantidad de dólares ingresada deberá ser sumada al Total_ Año a la fecha.
Este valor debe ser positivo.
Hay tres condiciones de error que producen un mensaje. El dato es no numérico, tiene un valor negativo o excede el valor máximo.
Si el monto es superior a US$100.000 se debe rechazar la transacción.
Al final del proceso se debe actualizar la base de datos con el total anual calculado.
Si todo estuvo correcto, imprimir totales de control y terminar.

Nota: Las operaciones en UF se tratan de manera similar a como se especificó para los dólares.

Inspección de requerimientos

Ejercicio Nº 3

Al aplicar la lista de palabras ambiguas se encuentran los siguientes problemas:

La cantidad de dólares ingresada **deberá** ser sumada al **Total_ Año** a la fecha. **Este valor debe** ser positivo.
Hay tres condiciones de error que **producen** un mensaje. El **dato** es no numérico, tiene un **valor** negativo o excede el **valor máximo**.
Si el monto es superior a US$100.000 se **debe** rechazar la transacción.
Al final del proceso se **debe actualizar** la base de **datos** con el total **anual calculado**.
Si **todo** estuvo correcto, imprimir totales de control y terminar.

Nota: Las operaciones en UF se tratan de manera **similar** a como se especificó para los dólares.

El problema de las interpretaciones

Lea la siguiente historia:

Un comerciante acaba de encender las luces de una zapatería, cuando de pronto surge un hombre pidiendo dinero. El propietario abre una máquina registradora. El contenido de la máquina registradora es retirado y el hombre corre. La policía es inmediatamente avisada.

El problema de las interpretaciones

Declaraciones acerca de la historia:

1.	Un hombre aparece apenas el propietario enciende las luces de la zapatería.	V	F	S/I
2.	El ladrón era un hombre	V	F	S/I
3.	El hombre no pidió dinero	V	F	S/I
4.	El hombre que abrió la máquina registradora era el propietario de la máquina	V	F	S/I
5.	El propietario de la zapatería retiró el contenido de la máquina registradora y huyó	V	F	S/I
6.	Alguien abrió una máquina registradora	V	F	S/I
7.	Después que el hombre que pidió el dinero tomó el contenido de la máquina registradora , huyó	V	F	S/I
8.	Aún cuando había dinero en la máquina registradora, la historia no dice cuanto	V	F	S/I
9.	El ladrón pidió dinero al propietario	V	F	S/I
10.	La historia registra una serie de acontecimientos que involucran a 3 personas: el propietario, un hombre que pidió dinero, y un miembro de la policía	V	F	S/I
11.	Los siguientes acontecimientos de la historia son verdaderos: alguien pidió dinero, una máquina registradora fue abierta, su dinero fue retirado, y un hombre de huyó de la tienda	V	F	S/I

Inspección de requerimientos

CHECKLIST INFORME DE ANALISIS DE REQUERIMIENTOS

SISTEMA	
MODULO	
FECHA INICIO	
FECHA TÉRMINO	
ENCARGADO	

	INFORME DE REQUERIMIENTOS	

1	ASPECTOS GENERALES	Tipo Error
1.1	El informe de Requerimientos utiliza las definiciones y formas del MDP.	
1.2	Es posible relacionar cada requerimiento con uno o más objetivos del Proyecto.	
1.3	Están todos los requerimientos completos y claramente identificados	
1.4	Existen Actividades de Seguridad.	
1.5	Existen Actividades de Respaldo.	
1.6	Existen Actividades de Recovery ante fallas	
1.7	Métodos de auditoría	
1.8	Criterios de medición y evaluación del desempeño del sistema (plan de calidad)	
1.9	Existen Actividades de Contingencia.	
1.10	Existen Actividades de Manejo de Información Histórica.	

2	MODELO DE DATOS

2.1	El Modelo de Datos ha sido aprobado por Administración de Datos.	
2.2	El Modelo de Datos utiliza las definiciones y especificaciones del MDP.	
2.3	Es posible navegar por el Diagrama Entidad-Relación para resolver cada uno de los requerimientos	
2.4	Las Entidades y sus Atributos (del diag E-R) son los necesarios (ni más ni menos) para satisfacer todos los requerimientos.	

Inspección de Código

Inspección de código

Objetivos específicos

- Detectar defectos de programación
- Asegurar que la codificación cumple con los requisitos de diseño
- Asegurar que la codificación cumple con los estándares de programación de la organización
- Determinar que el programa está apto para ser probado

Inspectores

- Moderador
- Autor del programa
- Analista que diseñó el programa
- Un experto en pruebas

Inspección de código

Elementos de entrada

- Listado de compilación sin errores
- Opciones de compilación útiles: Cross-reference, listado de variables, etc.
- Diccionario de datos del sistema
- Listado de defectos de inspecciones anteriores del programa

Inspección de código

CHECKLIST VISUAL BASIC	
ITEM	**CONTROL**
Generales	
1	Si una propiedad es pública no debe tener asociada una property.
2	Si una propiedad es privada siempre debería ser accesada a través de las property.
3	La property (get o let) tienen una regla asociada
4	Las propiedades de las clases están inicializadas en el evento de inicialización.
5	Las propiedades y métodos deben estar orientados a la identificación de la clase.
6	Las funciones y métodos asociados tienen relación entre ellos.
Particulares	
7	Se debe usar OPTION EXPLICIT.
8	Existen variables con nombres similares que puedan inducir a error.
9	Las variables declaradas fuera de Rutinas, Funciones, o Eventos siempre son definidas como Private y/o Public.

Inspección de código

Nº	Lugar	Observaciones	No Ch/List	Estado
1	Num_Transaccion	Se utiliza código duro en instrucción SQL. querytxt = "insert into qs36f.mtrnum00 (NUMTRA) values (**1**)" ***(2)*** *El nombre de la constante no tiene asociado un significado.*	23	ERR
2	Log	Variable **num_transac** sin tipo definido	14	OK
3	Log	Variable **correlativo** sin tipo definido	14	OK
4	Log	Conversión implícita de variable **fecha_transaccion** en la construcción de sentencia sql ***(2)*** *(fecha_transaccion es de tipo string ¿se valida el formato anteriormente?)*	13	NC
5	Log	Variable **hora_transaccion** sin tipo definido	14	OK
6	Log	Variable **monto** sin tipo definido	14	OK
7	Log	Variable **tipo_movimiento** sin tipo definido	14	OK
8	Log	Variable **estado_transac** sin tipo definido	14	OK
9	Log	Variable **Rut** sin tipo definido	14	OK

Walk Throughs

Los WalkThroughs son una técnica muy parecida a las inspecciones, pero, que es más informal y requiere menos recursos.

Características de los WalkThroughs

1. Es realizada por los “pares”
2. No requiere un moderador
3. La revisión es dirigida por el autor del programa
4. No se usa checklist de errores
5. No se registran estadísticas
6. El interesado es quien se encarga de hacer seguimiento y corrección de los defectos
7. No hay actividades tendientes a la prevención de defectos
8. No hay análisis organizacional de los resultados

WalkThroughs

Técnica de realización de los WalkThroughs

El autor analiza el producto con sus compañeros quienes le hacen preguntas y sugerencias. Las modificaciones al producto son de responsabilidad del autor.

Beneficios de los WalkThroughs

1. Bajo costo de realización
2. Los errores se detectan oportunamente
3. Aumento de la calidad del producto

Resultados obtenidos con Inspecciones

Nótese que el costo de encontrar errores en inspección tiene un promedio aproximado de 0.5 horas, mientras que a través de las pruebas tiene un costo promedio de 8 horas.

Por lo tanto, todos los esfuerzos que se hagan con las inspecciones tienen una alta rentabilidad para el proyecto y la empresa.

Una buena inspección de código puede ser equivalentea 30.000 casos de pruebas.

Inspecciones

Horas para encontrar un defecto

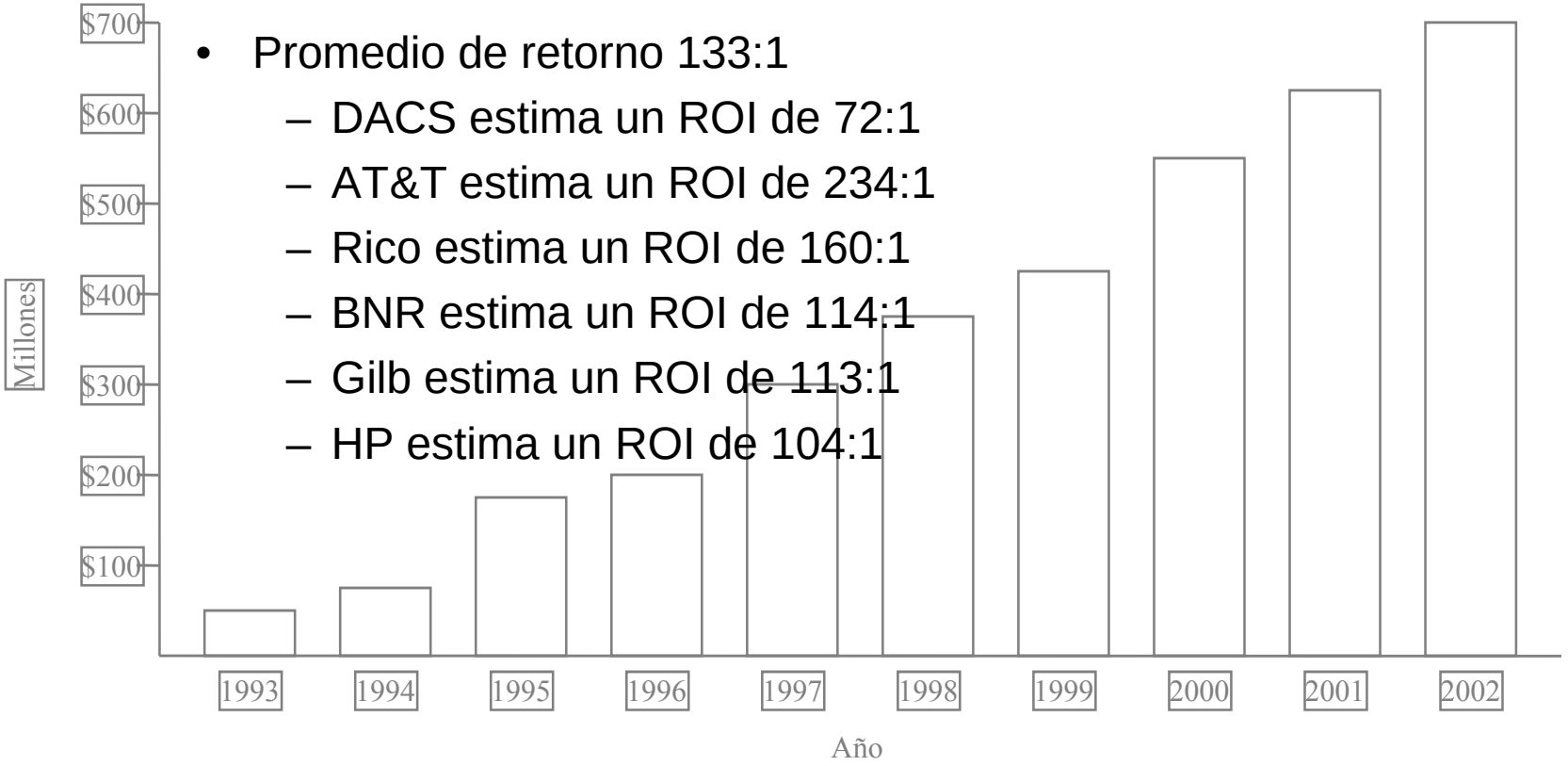
Autor	Pruebas 1	Inspección 2-10
Ackerman	0.26	
O'Neill		20
Russell	1	2-4
Shooman	0.6	3.05
Van Genuchten	0.25	8
Weller	0.7	6

Fuente: American Programmer.

Inspecciones

Retorno de la inversión

- Promedio de retorno 133:1
  - DACS estima un ROI de 72:1
  - AT&T estima un ROI de 234:1
  - Rico estima un ROI de 160:1
  - BNR estima un ROI de 114:1
  - Gilb estima un ROI de 113:1
  - HP estima un ROI de 104:1



Incremento de la calidad

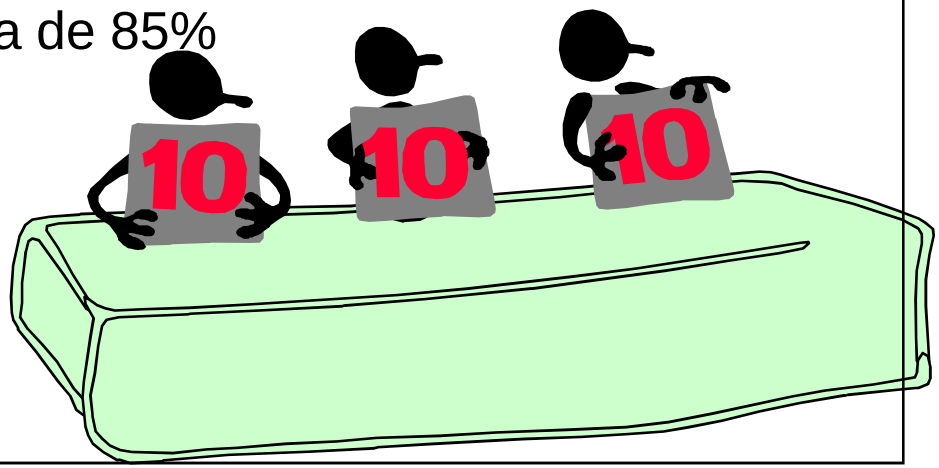
- Promedio de incremento de la calidad 16.4x
  - Rico estima un incremento de 3.03x
  - Bull HN estima un incremento de 76.93x
  - Aetna estima un incremento de 5.56x
  - IBM estima un incremento de 14.29x
  - BNR estima un incremento de 4.99x
  - AT&T estima un incremento de 3.35x
  - Fagan estima un incremento de 6.67x

Aumento de productividad

- Promedio de aumento de productividad de 6x
  - DACS estima un incremento de 1.55x
  - Fagan estima un incremento de 6.67x
  - AT&T estima un incremento de 8.37x
  - Rico estima un incremento de 6.54x
  - BNR estima un incremento de 5.17x
  - Gilb estima un incremento de 5.13x
  - HP estima un incremento de 4.84x

Eficiencia en remoción de defectos

- Promedio de eficiencia de remoción 82.2%
  - Rico estima una eficiencia de 67%
  - Bull HN estima una eficiencia de 98.7%
  - Aetna estima una eficiencia de 82%
  - IBM estima una eficiencia de 93%
  - BNR estima una eficiencia de 80%
  - AT&T estima una eficiencia de 70%
  - Fagan estima una eficiencia de 85%



Otros resultados obtenidos

- AT&T la evaluó en un proyecto 20 veces más efectiva que la prueba con computador.
- AETNA INSURANCE Co. encontró el 82% de los errores en un programa Cobol.
- TRW analizó que de un total de 2019 problemas, un 62.7 % podrían haber sido detectados con Walk-Through
- Watt Humprey reporta que entre 60% a 80% de los defectos son encontrados con las inspecciones.
- América XXI revisó 15.000 LOC en producción y encontró 2.019 defectos, 48 de los cuales requerían corrección urgente.

Inspecciones

Cabe preguntarse a esta altura,

¿Por qué las empresas no introducen las inspecciones en sus procesos?.

Humphrey dice: “Hay dos razones”.

“1. Las planificaciones son basadas en suposiciones, y esas suposiciones son, casi siempre irreales. Luego, la presión hace que los desarrolladores vayan de crisis en crisis y nadie tenga tiempo de pensar en nada más que probar, “debuguear” y arreglar.

2. No existe una disciplina de recolectar datos, por lo tanto, los desarrolladores no conocen el número de defectos que existen ni el costo que tiene el encontrar y arreglar esos defectos. Así, ellos raramente aprecian el enorme costo que puede evitarse encontrando y arreglando esos defectos antes de las pruebas.”

Porqué no se usan las Inspecciones?

- No están en el curriculum de estudios
- Falta de buena literatura al respecto
- Inspecciones se transmiten informalmente
- Mucha gente las critica sin conocerlas
- Los benefícios son poco conocidos

Inspecciones

Cuáles son los obstáculos más comunes ?

- Los ingenieros sólo están interesados en diseño
- Usar la última tecnología es la prioridad más alta
- Los Administradores no conocen los beneficios
- Los Administradores no perciben el valor de la técnica
- La fecha de término tiene prioridad por sobre la calidad y confiabilidad del producto
- Simplemente, no me tinca

Inspecciones

Cuáles son los mitos más comunes?

- Es demasiado caro
- Es una técnica obsoleta de la era de los mainframe
- No es aplicable a las tecnologías de la era de la Internet
- No es más efectivo a que el propio desarrollador revise su trabajo

Inspecciones

Cuales son los errores más comunes?

- Atacar o cuestionar al autor
- No tomarlas en serio
- Inspeccionar a alta velocidad
- Inspeccionar por más dos horas seguidas
- No preparar las inspecciones
- No entrenar ni practicar lo suficiente
- Resolver los errores durante la inspección

Introducción de la Inspecciones en su empresa

Actividades iniciales

1. Conseguir el respaldo gerencial.
2. Entrenar al personal de desarrollo y de pruebas.
3. Designar y entrenar al moderador.
4. Identificar los métodos formales existentes para elaborar los productos.
5. En caso que no haya estándares, estos deben ser definidos.
6. Confeccione y distribuya un checklist de revisión.
7. Capacitar, capacitar, capacitar.

Introducción de las Inspecciones en su empresa

Selección de proyecto piloto

1. Seleccione un proyecto piloto.
2. Debe ser importante, pero, no crítico.
3. Las inspecciones a realizar deben hacerse sobre aquellas fases en que haya estándares definidos y ojalá internalizados por el equipo de desarrollo.
4. El proyecto debe cumplir con sus fases a inspeccionar en un tiempo corto. Inferior al promedio de la organización.
5. Los involucrados deben ser proclibes a la metodología.
6. Debe asignarse los recursos necesarios y el tiempo para estas actividades debe ser planificado.
7. Debe existir la decisión gerencial de usar la técnica sin claudicaciones.

Inspecciones

Algunos aspectos importantes a tener en cuenta

1. La gerencia no debe usar la cantidad de errores encontrados como medida negativa de evaluación del autor.
2. Un mayor número de errores encontrados, generalmente, significa una mejor inspección y no un producto más malo.
3. Si los errores son percibidos como un signo negativo, habrá tendencia a ocultarlos o a retrasar el momento de la inspección.
4. La calidad del trabajo de una persona se mide por el resultado final.
5. Los inspectores comparten la responsabilidad por la calidad del producto final.

Inspecciones

Dada la naturaleza intuitiva de este método, es conveniente mencionar que las inspecciones constituyen uno de los métodos más efectivos en el proceso de prueba de software. La mayoría de las empresas que necesitan producir software de alta calidad lo consideran como una parte fundamental de este proceso (NASA, IBM entre otras).

Técnicas de Validación

Enfoques de las Técnicas de Validación

En las técnicas para pruebas se distinguen dos enfoques:

- **Caja negra**

Este enfoque está orientado a la construcción de casos de prueba a partir de las funcionalidades del sistema. El software es visto como una “caja negra” que ante ciertas entradas debe producir determinados resultados. En esta categoría se ubican las técnicas de :

- Clases de Equivalencia
- Análisis de Valores Límites
-Causa y Efecto

- **Caja Blanca**

Este enfoque está orientado a la construcción de casos de pruebas a partir del código del programa. Los casos que se construyen están destinados a probar el funcionamiento de los ciclos iterativos, variables, rutinas, caminos, etc. En esta categoría se ubica las técnicas de:

- Cobertura de Instrucción
- Cobertura de Decisión

Enfoques para la Prueba de Componentes

La Prueba de Componentes es una prueba orientada a validar la correctitud de los distintos módulos que contiene un sistema a medida que se va avanzando en su construcción. Existen dos estrategias básicas:

Top Down

Es una prueba descendente de las componentes. Para ello es necesario desarrollar módulos “dummies” que corresponden a aquellos módulos de menor nivel jerárquico que aún no se construyen.

Bottom up

Es una prueba ascendente de las componentes. Para ello es necesario desarrollar módulos “drivers” que corresponden a aquellos módulo de nivel jerárquico superior que aún no se construyen.

Modelo Transaccional

Modelo Transaccional

El Modelo Transaccional permite apoyar al especialista en la tarea de aplicar las técnicas a un sistema de software específico.

Su utilización permite transformar la tarea de pruebas de una aplicación computacional en un conjunto estructurado de pasos.

Modelo Transaccional

Pasos a seguir

1. **Identificar los módulos de la aplicación**

   Ej: Procesos, Requerimientos, Casos de Uso, Opciones del menú, etc.

2. **Identificar los programas o aplicaciones**

   Ej: Validador, Actualizador,...

3. **Identificar las transacciones de cada módulo**

   Ej: Solicitar precio, Comprar, Cálcular Comisión, Confirmar Pedido, Convertir moneda, Depositar, Liquidar, Facturar, etc.

4. **Registrar el análisis transaccional**

   Registrar la información de Datos de Prueba en Formulario.

5. **Definir Importancia de la transacción**

   En base al juicio de un experto (usuario)asignar una importancia a cada transacción

Modelo Transaccional

6. Priorizar módulos y transacciones

Hacer una lista por orden de importancia

7. Analizar las características de transacciones a probar

Ej: Rangos, límites, ...

8. Definir Casos de Prueba

Para cada transacción especificar los casos/datos de prueba

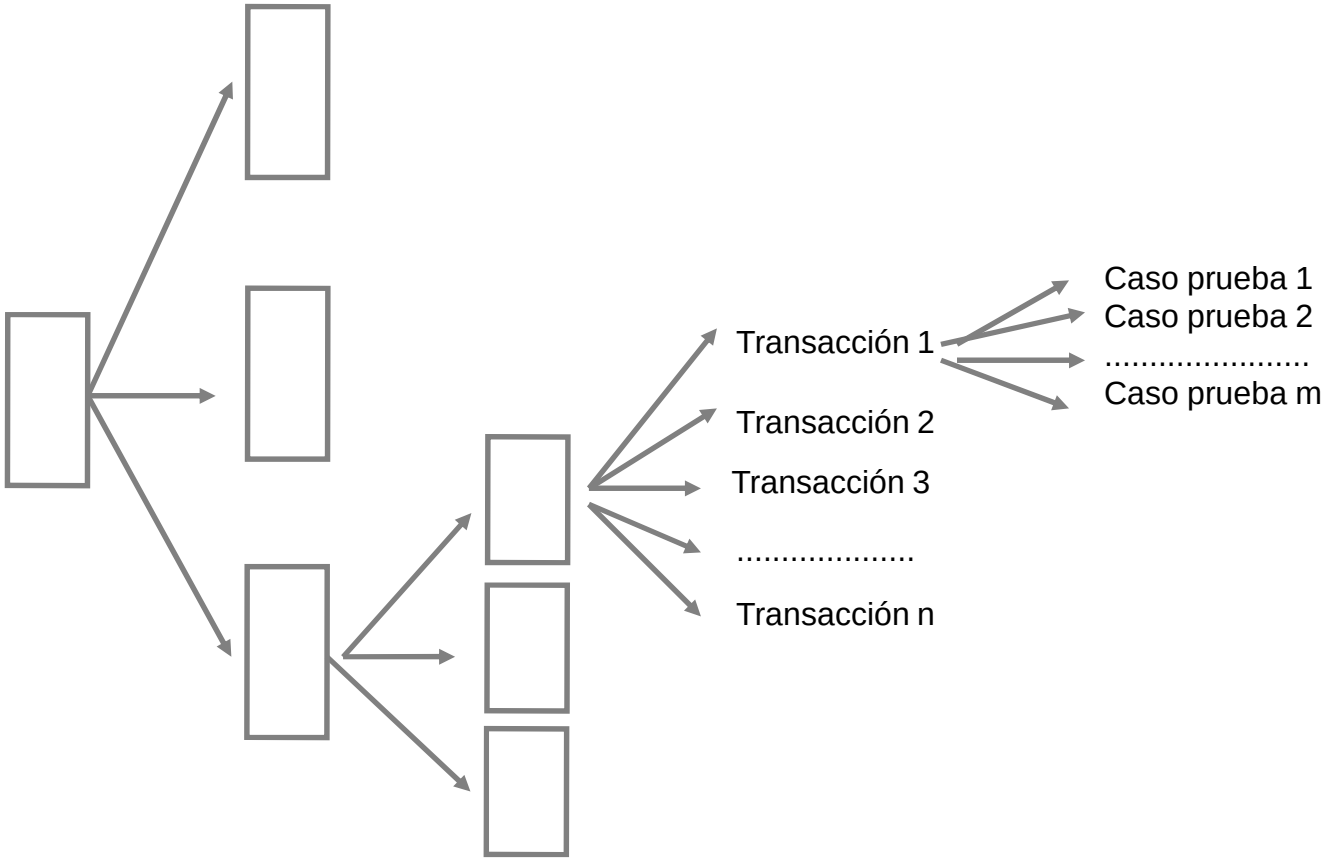
9. Registrar Datos de Prueba

Registrar la información de Datos de Prueba

Análisis Transaccional

Nivel principal	Nivel Pgm.	Nivel Tr.	NivelCasoPrueba
Proceso Nº1 Sub proc. 1.1 Sub proc. 1.2 Sub proc. 1.3	Comp. 1 Comp. 2	Tr 1 Tr 2	Caso 1 Caso n

Análisis Transaccional



Análisis Transaccional y Escenarios de Negocios

Un Escenario de Negocios es un conjunto de acciones que están relacionados con una operación del sistema.

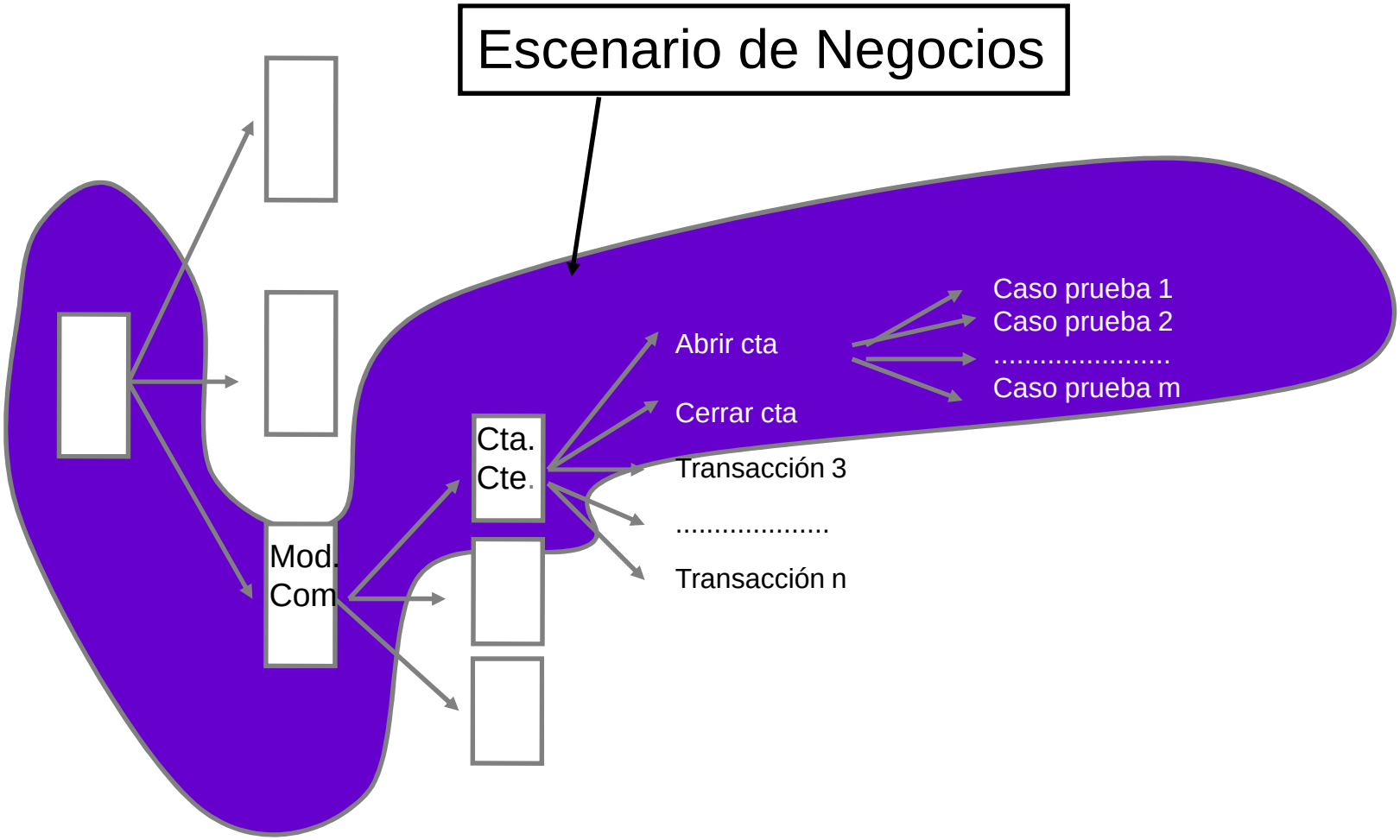
Ejemplos de Escenarios de Negocios:

-Administrar una Cuenta Corriente
-Otorgar un Crédito
-Registrar una venta

En muchas ocasiones es necesario probar un Escenario de Negocios como una unidad completa. Para esto se identifican las transacciones individuales que están involucradas el el Escenario y se desarrollan los casos de pruebas para cada una de las distintas situaciones. Todas ellas se agrupan como un Caso de Prueba de Escenario y se aplican integramente. Esto puede incluir acciones interactivas y/o Batch.

Modelo Transaccional

Análisis Transaccional y Escenarios de Negocios



Análisis Transaccional y Escenarios de Negocios

De esta forma, cuando se utiliza esta técnica, tenemos que el número de Casos de Pruebas puede ser muy similar al número de Escenarios de Negocios definidos.

Sin embargo, un análisis más fino nos muestra que al interior de un Caso de Prueba de un Escenario de Negocios existen muchos casos de pruebas de transacciones individuales.

Por lo tanto, es necesario llevar un control detallado y ponderado de la situación de cada Caso de Prueba de Negocio, ya que éste involucra a múltiples casos de pruebas individuales.

Clases de Equivalencia

Clases de Equivalencia

- Es una técnica para generación de casos de pruebas.
- Se basa en los siguientes principios:

★Reduce significativamente el número de otros casos de prueba. Es decir, cada caso invoca el mayor número de condiciones de entrada diferentes a fin de minimizar el número total de casos necesarios.

★Cubre un conjunto extenso de otros casos de prueba posibles. Es decir, si un caso de prueba perteneciente a una clase de equivalencia detecta un error, se espera que cualquier otro caso de prueba perteneciente a la misma clase detecte el mismo error. Inversamente, si un caso de prueba no detecta un error, no se espera que otros casos de prueba de la misma clase puedan detectarlo.

Metodología de Clases de Equivalencia

• La metodología de Clases de Equivalencia consta de dos pasos:

★Identificación de las Clases de Equivalencia

★Definición de los Casos de Prueba

★Identificación de las Clases de Equivalencia

- ✧ Se toma cada condición de entrada de la especificación y se divide en dos o más grupos
- ✧ Se identifican las clases válidas (entradas válidas) y las inválidas (valores de entrada “erróneos”)
- ✧ La identificación de las clases es, en gran medida, un proceso intuitivo

Algunas pautas se describen a continuación:

Clases de Equivalencia

★ Identificación de las Clases de Equivalencia

1. Si una condición de entrada especifica un rango de valores (por ej., “La numeración de un ítem puede ser de 1 a 999”, identificar una clase de equiv. válida (1< Número de ítem< 999) y dos inválidas (Número de ítem< 1 y número de ítem> 999).

2. Si una condición especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos (P. ej., “el tipo de vehículo debe ser BUS, CAMION, TAXI, CA MIONETA, MOTOCICLETA”, identificar una clase equivalencia válida para cada uno de ellos y una clase inválida (P.ej. TRAILER).

3. Si una condición de entrada especifica una situación del tipo “debe ser (P.ej. “el primer carácter del identificador debe ser una letra”), identificar una clase válida (es una letra) y una inválida (no es una letra).

4. Si existe alguna razón para creer que ciertos elementos pertenecientes a una clase de equivalencia no son tratados en forma idéntica por el programa, dividir la clase de equivalencia en clases de equivalencia menores.

★ Identificación de los Casos de Prueba

El proceso de identificación de los casos consiste en :

1. Asignar un número único a cada clase de equivalencia

2. Hasta que todas las clases de equivalencia válidas hayan sido cubiertas por casos de prueba, escribir un nuevo caso de prueba que cubra tantas clases de equivalencia válidas, no cubiertas, como sea posible.

3. Hasta que todas las clases de equivalencia inválidas hayan sido cubiertas por casos de prueba, escribir un caso de prueba para cubrir una y sólo una, de las de las clases de equivalencia no cubiertas.

La razón para cubrir los casos inválidos mediante casos de prueba individuales es que ciertos controles de entrada enmascaran o invalidan otros controles similares. Por ej., si la especificación dice”entrar tipo de libro (TELA, CARTON, PAPEL), y cantidad (1- 9999)”, el caso de prueba “XYZ 0” que contiene dos condiciones de error (Tipo de libro y cantidad inválidos), probablemente, no detectará el error en la cantidad y el programa diga sólo, “tipo de libro inválido”.

Ejemplo de Clases de Equivalencia

CLASES VALIDAS		CLASES INVALIDAS	
Fecha1 > 1.1.1990	(1)	Fecha1<=1.1.1990	(6)
1.3.1986<Fecha2<Hoy	(2)	Fecha2<=1.3.1986	(7)
		Fecha2>=Hoy	(8)
1.3.1982<Fecha3<2.1.1990	(3)	Fecha3<=1.3.1982	(9)
Fecha3=2.1.1990	(4)		
2.1.1990<Fecha3<30.3.1996	(5)	Fecha3>= 30.3.1996	(10)

Ejemplo de Clases de Equivalencia

CASOS DE PRUEBA

	Fecha1	Fecha2	Fecha3	
1.	05.03.1997	01.10.1988	10.10.1985	(1),(2),(3)
2.	05.03.1997	01.10.1988	02.01.1990	(4)
3.	05.03.1997	01.10.1988	15.08.1993	(5)
4.	08.04.1988	01.10.1988	15.08.1993	(6)
5.	05.03.1997	15.07.1984	15.08.1993	(7)
6.	05.03.1997	27.11.2010	15.08.1993	(8)
7.	05.03.1997	01.10.1988	29.05.1980	(9)
8.	05.03.1997	01.10.1988	07.12.1998	(10)

Ejemplo de Clases de Equivalencia

Analice el siguiente ejemplo:

Una aplicación calcula la prima de seguro a pagar por un vehículo en cada región del país. El único dato que se debe ingresar es la Región.¿Cuantos casos de prueba haría Ud. en cada una de las siguientes reglas de negocio?

P = Tasación * 0.25% en todas las regiones (1a a 13a)

P = Idem anterior pero P = (Tasación * 0.25%) - 0.1% por cada año de antigüedad en la 1a y 12a región

P = Tasación * 0.25% en todas las regiones pero no hay cobertura en la 4a region

P = Tasación * Factor % en todas las regiones. Factor está en una tabla y es distinto para cada región

Análisis de Valores Límites

Análisis de Valores Límites

El análisis de valores límites es una técnica complementaria a las clases de equivalencia. La experiencia muestra que los casos de prueba que ejercitan situaciones límites producen mejores resultados que aquellas que no lo hacen. Las condiciones límites son las situaciones que se hallan directamente, justo en los bordes y arriba o debajo de los márgenes de las clases de equivalencia de entrada y de salida.

El análisis de valores límites se basa en dos principios básicos:

1. Se debe seleccionar uno o más elementos, de tal manera que cada margen de la clase de equivalencia quede sujeto a una prueba.

2. Se deben considerar las condiciones de entrada (espacio de entrada) y también el espacio de resultados, es decir, las clases de equivalencia de salida.

Para realizar el análisis de valores límites no es posible entregar recetas.
A continuación se entregan algunas reglas generales.

Reglas generales

1. Si una condición de entrada especifica un rango de valores, escribir casos de prueba para los extremos del rango, y casos de prueba inválidos para situaciones que están precisamente más allá de los extremos. Por ejemplo, si el dominio válido de los valores de entrada es -1.0 y 1.0, escribir casos de prueba para las situaciones -1.0, 1.0, -1.001 y 1.001.

2. Usar la regla 1 para cada condición de salida. Por ejemplo, si una aplicación calcula un impuesto mensual cuyo mínimo es $0,0 el máximo es $100.000,0 escribir casos que causen impuestos de $0,0 - $100.000,0. Ver también si es posible generar impuestos negativos y un impuesto superior a $100.000,0.

3. Analizar los máximos y mínimos almacenamientos posibles en las variables. (valores 0 y 99999).

4. Analizar la máxima y mínima cantidad de registros en los archivos. Incluso se debe considerar el archivo vacío.

Análisis de Valores Límites

El análisis de valores límites permite detectar una gran cantidad de errores comunes que se cometen en la construcción de software. Sin embargo, esos errores no son detectados si no se realizan casos de pruebas específicamente orientados a ellos.

El análisis de valores límites a menudo se lo emplea en forma inefectiva porque esta técnica, parece a simple vista, simple. Sin embargo, las condiciones límites son a menudo muy sutiles y su identificación requiere una cuidadosa elaboración.

Ejercicio Nº 4

Un instituto Pre Universitario desea poner a disposición de sus alumnos una aplicación que corra en Internet y que permita a los alumnos realizar ensayos de PAA y Pruebas Específicas. Están disponibles múltiples facsímiles para los distintos tipos de pruebas.

La aplicación consta de los siguientes módulos:

Módulo Administrador

Este módulo permite al administrador de la aplicación crear plantillas de pruebas en la BD. Para cada prueba se ingresan los siguientes datos:
-Código de la Prueba
-Nombre de la Prueba
-Número de preguntas de la Prueba
-Respuesta correctas para cada pregunta

Análisis de Valores Límites

Ejercicio Nº 4

Módulo usuario

Este módulo permite a un alumno realizar una ensayo de prueba.
Para realizar un ensayo, la aplicación debe solicitar al alumno ingresar su código de identificación (un número de 6 dígitos). Despliega una ListBox con los nombres de las pruebas disponibles. El alumno debe seleccionar una.

La aplicación en el lado del usuario debe validar que el código de identificación sea numérico y que se haya seleccionado un tipo de prueba. Si hay error vuelve a solicitar los datos. Esto hasta un máximo de 3 veces. Si se sobrepasa este número, se cancela la aplicación.

Si esta correcto, se activa una componente que despliega un formulario con las primeras 5 preguntas del examen. Una vez que el alumno aprieta la tecla Enter o hace Click en Ingresar, los datos del formulario deben ser almacenados en la Base de Datos de pruebas por revisar.

Se permite sólo un alumno a la vez responder cada prueba. Si ya hay un alumno respondiendo una prueba, a un segundo alumno se debe dar un mensaje que intente dentro de una hora más.

Para cada pregunta hay un tiempo máximo de 5 minutos y para la prueba completa un tiempo máximo de 2.5 horas. Cualquiera de estas dos condiciones que se sobrepase, se debe desconectar al usuario.

Nota: Con fines didácticos este ejemplo contiene simplificaciones en relación a un caso real.

Análisis de Valores Límites

Ejercicio Nº 4

Tabla Pruebas

Contiene registros de 60 caracteres.
Los tres primeros caracteres son un número que corresponde al código correlativo de la prueba. (un valor comprendido entre 1 y 200).
Los campos 4 al 6 contienen el número de preguntas de la prueba.
Los campos 10 a 50 contienen el nombre del examen cuyo contenido se utiliza para desplegar en una ListBox y como título en cada informe de salida

Tabla Respuestas Correctas

Contiene registros de 60 caracteres.
Las columnas 1-3 contienen el código de la prueba.
Las columnas 10-59 contienen las respuestas correctas para las preguntas 1-50 (sólo caracteres de la “a” a la “e”). Los registros siguientes contienen, en las columnas 10-59, las respuestas correctas para las preguntas 51-100, 101-150, y así hasta terminar.

A continuación vienen los registros con respuestas correctas para la prueba siguiente.

Análisis de Valores Límites

Ejercicio Nº 4

Tabla Respuestas de Alumnos

Contiene registros de 15 caracteres.

Cada registro contiene en las columnas 1-6 el código del alumno.
Las columnas 7-9, contienen el código de la prueba.
Las columnas 11-15 contienen las respuestas del alumno para las preguntas 1-5.
Si el examen consistiese de más de 5 preguntas, los registros siguientes para el alumno contendrían en las columnas 11-15 sus respuestas a las preguntas 6-10, 11-15, etc.

Suponga que sólo existe un alumno y una prueba en esta tabla.

Esta información se gráfica de la siguiente manera:

Tabla de Pruebas

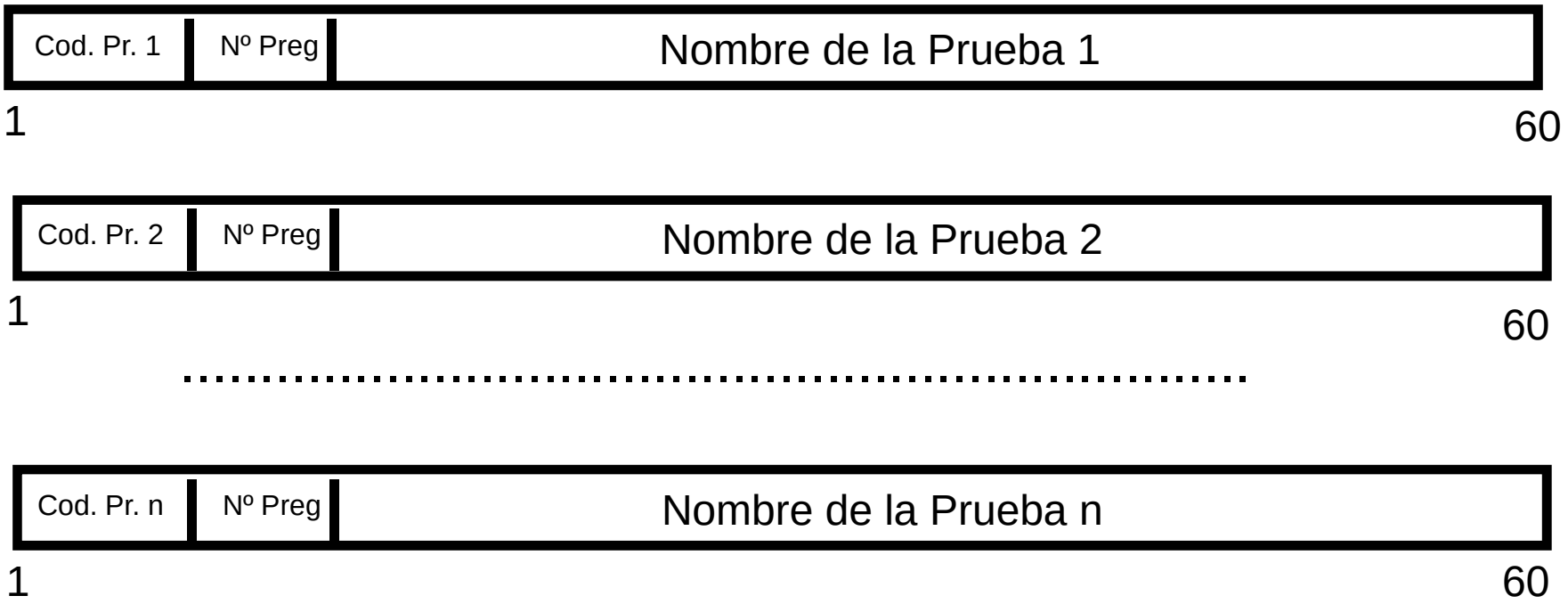


Tabla de Respuestas Correctas

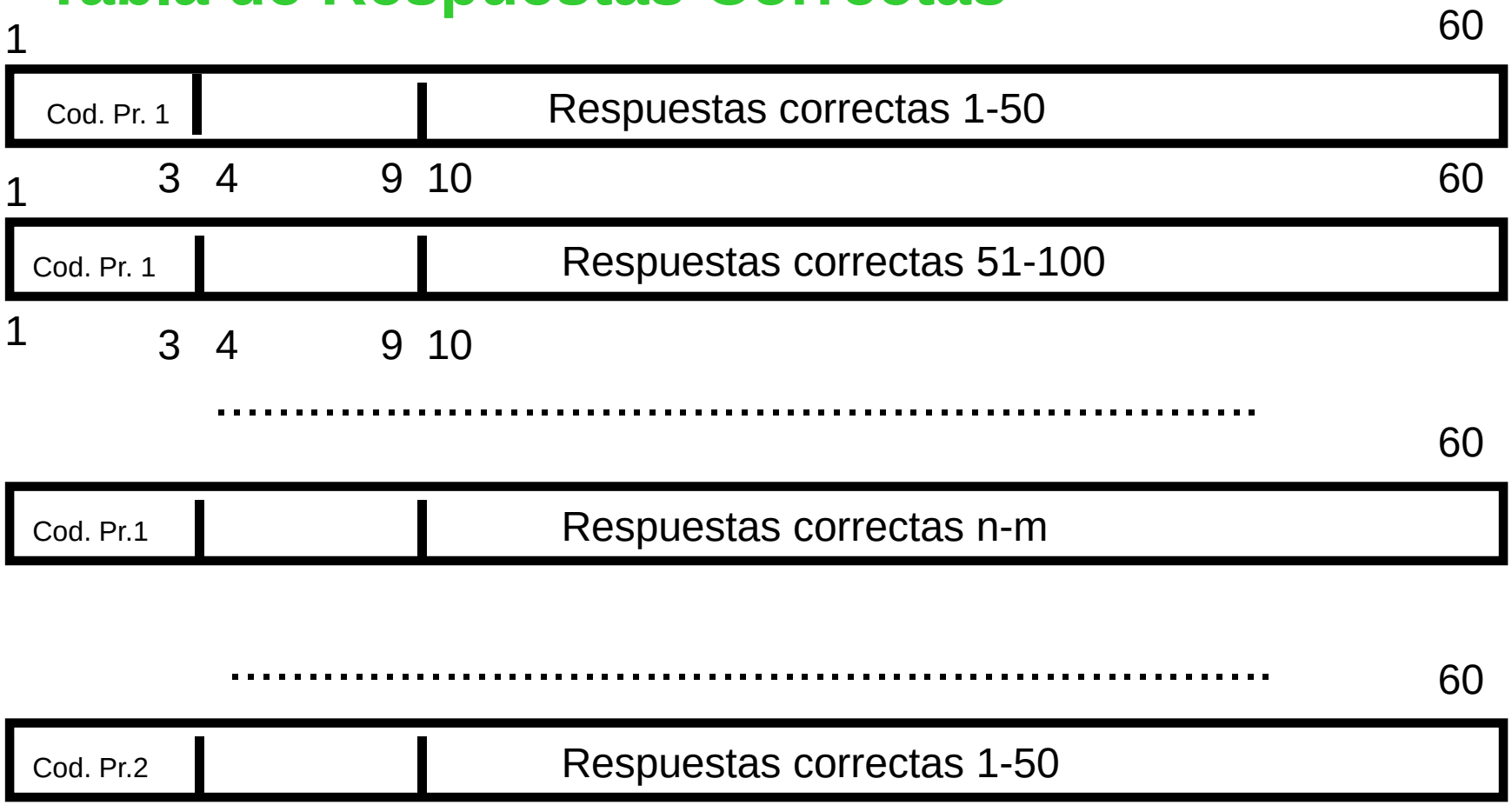
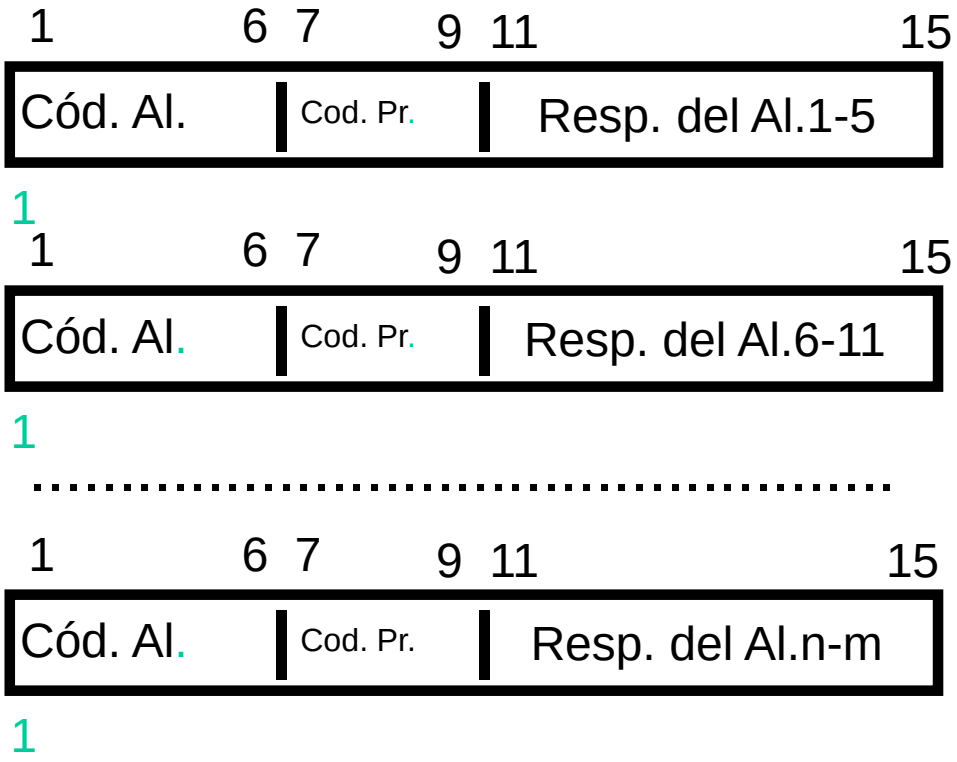


Tabla de Respuestas del Alumno



Análisis de Valores Límites

La aplicación genera las siguientes salidas:

1. Muestra las respuestas del alumno y las respuestas correctas, la calificación obtenida por el alumno (porcentaje de respuestas correctas) y orden comparado con los últimos 50 alumnos de la misma prueba.

2. La segunda salida es similar a la anterior, pero ordenada por código del alumno.

3. La tercera salida contiene el promedio, la mediana y la desviación estándar de las calificaciones de los últimos 50 alumnos.

4. La última salida es ordenada por número de pregunta, que muestra el porcentaje de alumnos que respondieron correctamente a cada una de ellas.

Análisis de Valores Límites

Casos de prueba recomendables al Módulo Administrador

Relativos al número de registros de la Tabla de Pruebas

1. Tabla con 0 registros
2. Tabla con 1 registro
3. Tabla con 200 registros
4. Tabla con 201 registros

Relativos al código de la prueba

5. Una prueba con Código = 0
6. Una prueba con Código = 1
7. Una prueba con Código = 200
8. Una prueba con Código = 201
9. Una prueba con Código = 999

Relación entre registros y código (cantidad de pruebas)

10. 1 registro menos que lo indicado en código más alto
11. = cantidad de registros que lo indicado en código más alto
12. 1 registro más que lo indicado en código más alto

Análisis de Valores Límites

Casos de prueba recomendables al Módulo Administrador

Relativos a la Tabla de Respuestas Correctas

13. Una Prueba con 0 pregunta
14. Una Prueba con 1 pregunta
15. Una Prueba con 50 preguntas
16. Una Prueba con 51 preguntas
17. Una Prueba con 999 preguntas
18. Una Prueba con 1000 preguntas

19. Tabla con 0 pruebas
20. Tabla con 1 pruebas
21. Tabla con 200 pruebas
22. Tabla con 201 pruebas

Relación entre tabla de Pruebas y Tabla de Respuestas Correctas

23. 1 Prueba menos en Respuestas Correctas que en Pruebas
24. = cantidad de Pruebas en Respuestas Correctas que en Pruebas
25. 1 Prueba más en Respuestas Correctas que en Pruebas
26. 1 respuesta correcta menos que lo indicado para esa prueba
27. = número de respuestas correctas que lo indicado para esa prueba
28. 1 respuesta correcta más que lo indicado para esa prueba

Análisis de Valores Límites

Casos de prueba recomendables al Módulo Administrador

Relativos a las contenidos de las Respuestas Correctas

29. Una respuesta < “a”
30. Una respuesta = “a”
31. Una respuesta = “e”
32. Una respuesta = “f”

Análisis de Valores Límites

Casos de prueba recomendables al Módulo Usuario

Validación en el lado del usuario

33. Código = 0
34. Código = 1
35. Código = 999999
36. Código = 1000000

37. Seleccionar 0 Prueba
38. Seleccionar 1 Prueba
39. Seleccionar dos Pruebas

40. Ingresar un error 2 veces
41. Ingresar un error 3 veces
42. Ingresar un error 4 veces

Nota: ¿Tienen sentido todos estos casos ?

Análisis de Valores Límites

Casos de prueba recomendables al Módulo Usuario

Despliegue de Formulario con Preguntas

43. Una Prueba con 0 pregunta
44. Una Prueba con 1 pregunta
45. Una Prueba con 5 preguntas
46. Una Prueba con 6 preguntas
47. Una prueba con 50 preguntas
48. Una Prueba con 999 preguntas
49. Una Prueba con 1000 preguntas

Relativos a la cantidad de alumnos intentando responder una prueba

50. 0 alumno (?)
51. 1 alumno
52. Segundo alumno intenta la misma prueba
53. 1 Alumno intenta responder la prueba anterior
54. 1 alumno intenta responder la prueba siguiente

Análisis de Valores Límites

Casos de prueba recomendables al Módulo Usuario

Relativo al tiempo de reintentar

55. Reintentar en 59 minutos más
56. Reintentar en 1 hora más
57. Reintentar en 61 minutos más

Relativo al tiempo de respuesta de cada pregunta

58. Responder a una pregunta a los 4’
59. Responder una pregunta a los 5’
60. Responder a una pregunta a los 6’

Relativo al tiempo de respuesta total de la Prueba

61. Responder toda la prueba a las 2h29’
62. Responder toda la prueba a las 2h30’
63. Responder toda la prueba a las 2h31’

Nota: ¿Que casos están de más en esta página ?

Análisis de Valores Límites

Casos de prueba recomendables relativos a las salidas 1 y 2. Algunos casos ya fueron cubiertos por casos anteriores.

64. 0 alumnos
65. 1 alumno
66. 50 alumnos
67. Todos los alumnos reciben la misma nota
68. Todos los alumnos reciben notas diferentes
69. Un alumno recibe una nota 0
70. Un alumno recibe una nota 100
71. Un alumno tiene el código de alumno más bajo posible
72. Un alumno tiene el valor más alto posible de código de alumno
73. El número de alumnos es tal que la extensión de la salida exige más de una pantalla (a fin de comprobar si hay despliegue anormal en la segunda pantalla)
74. El número de alumnos es tal que en una pantalla caben todos exactamente.

Análisis de Valores Límites

Casos de prueba recomendables relativos a las salidas 3 y 4

76. El valor medio es el máximo posible (todos los alumnos tienen el puntaje máximo)
77. El valor medio es 0 (todos los alumnos reciben un puntaje 0)
78. La desviación estándar tiene su valor máximo posible (la mitad recibe 0 y la otra mitad 100)
79. La desviación estándar es 0 (todos los alumnos reciben la misma nota)
80. El número de preguntas es tal que el informe de salida cabe en una pantalla
81. El número de preguntas es tal que el informe de salida las contiene a todas menos una en una pantalla.

Otros casos posibles

82. Todos los alumnos contestan correctamente la pregunta 1
83. Todos los alumnos contestan incorrectamente la pregunta 1
84. Todos los alumnos contestan correctamente la última pregunta
85. Todos los alumnos contestan incorrectamente la última pregunta

Nota: ¿Están todos las funciones cubiertas por al menos un caso de prueba?

Análisis de Valores Límites

Recomendaciones para los casos de Pruebas

1. Las modificaciones al background deben ser visibles. Para ello inicializar variables en 5 por ejemplo.

2. Las modificaciones al foreground deben ser visibles. Si el software deja una variable en 2, asegúrese que ella no está en 2 al comienzo.

3. Efectos alternativos deben causar diferencias visibles. Ejemplo:

```
If Trigger1 then
    Result is A+A
If  Trigger2 then
    Result is A*A
```

Supongase un caso de prueba que debe activar el Trigger1, si por error se activa el Trigger2, A=2 no detectará el error.

4. Ingresar valores distintos para distintas variables.
   Un error común en programación es usar las variables cambiadas. Por ej.:
   CALL Rutina (A;B) si se ingresa (1,1) no será posible detectar el error.

5. Ingreso y devolución de argumentos
   Si un programa hace una búsqueda de un carácter en un string y devuelve su posición
   SEARCH(Caracter, String, Posicion)
   Si Caracter=x; String=XYZ y
   Posicion está inicializado en 1, si la búsqueda no se realiza no se detecta el error.

Catálogo de Pruebas

Catálogo de Pruebas

Descripción

El Catálogo de Pruebas es un compendio de
casos de prueba que permiten probar una serie
de situaciones y elementos de una aplicación de software.

El Catálogo de pruebas puede ser de gran ayuda en las pruebas unitarias.

Ventajas

- Simplificar la selección de casos
- Mejorar la calidad de la prueba

Catálogo de Pruebas

Catálogo de Testing

1. BOOLOEAN	Bool			
• A (true)				
• B (false)				
• Un valor distinto de A o B				
	Cas			
2. CASES				
V1 puede ser cualquier valor de (C1, C2,....,C9)				
• V1 = cada valor posible				
• Un valor como C1-1 y C1+1 si los Cn son enteros				
3. NUMEROS	Num			
Comparación de Operadores *(epsilon es 1 para enteros)*	Co			
<, > =				
• V1=V2 – epsilon				
• V1=V2				
>, < =				
• V1=V2				
• V1=V2 + epsilon				
= =, !=				
• V1 = = V2				
• V1 != V2				
Números usados en Operaciones Aritméticas	Noa			
El código a testear realiza operaciones complejas (no sólo sumas y restas)				
V1 = 0				
V1 = epsilon				
V1 = -epsilon				
V1 = el valor más grande posible				
V1 = el valor más pequeño posible				

Catálogo de Pruebas

9. COLECCIONES GENERALES	Cgr
• Vacía	
• Un elemento	
• Más de un elemento	
• Llena	
• Elementos duplicados	
Operaciones con colecciones ordenadas	Oco
Colecciones con un primer y último elemento	
• Trabajar con el primer elemento	
• Trabajar con el último elemento	
Crear subconjuntos de colecciones	sc
• Sacar todos los elementos	
• Sacar todos los elementos menos uno	
• Sacar un elemento	
• No sacar ningún elemento (filtro vacío)	
Borrar elementos de una colección	Bec
• La colección tiene un elemento	
• La colección no tiene elementos	

Catálogo de Pruebas

17. ARCHIVOS	Arc			
Chequear si un archivo existe	Cav			
• Archivo existe				
• Archivo no existe				
Apertura de un archivo para lectura	Aal			
• Archivo no existe				
• Archivo existe y es leible				
• Archivo existe, pero no es leible				
Apertura de un archivo para escritura	Aae			
• Archivo no existe, pero puede ser creado				
• Archivo no existe, pero no puede ser creado				
• Archivo existe y es "escribible"				
• Archivo existe, pero no es "escribible"				

EEjemplo de Uso de Catalogo de Pruebas

SSe debe probar una mantención hecha al sistema de Comercio Exterior. La mantención afecta al programa DEXP, el cual es activado a través del submenú Administración de Declaraciones del Menú General.
EEl programa tiene dos funcionalidades, que son: Ingreso y Elimina.
LLa mantención realizada afecta a la funcionalidad Elimina y consiste en lo siguiente:

Se debe eliminar todas las Declaraciones cuyo número de identificación sea menor o igual a 1000.
Las Declaraciones eliminadas se deben grabar en el archivo Eliminados, el que puede o no contener declaraciones anteriores.

SSe pide hacer el Análisis Transaccional y los casos de pruebas de la mantención descrita.

FFormulario 4

Opción/Pgma	**Transacciones**
Menú Gral Adm. Declaraciones DEXP	- - - Ingreso (no) Elimina

Catálogo de Pruebas

FFormulario 5

Menu Gral
Ad. Decl.
Elimina[Num(co), Arc(Cav,Aal,Aae), Col(Cgr,Oco, Sc,Bec)]
Caso 1
Causas
Num. Dexp=1000
Efectos esperado: Eliminar Declaración
Para efectos del ejemplo sólo se anotarán las causas de los restantes casos :
Caso 2: Num. Dexp=1001
Caso 3: Archivo DEXP no existe
Caso 4: Archivo DEXP existe y es leíble
Caso 5: Archivo DEXP existe y no es leíble
Caso 6: Archivo DEXP vacío
Caso 7: Archivo DEXP con un elemento
Caso 8: Archivo DEXP más de un elemento
Caso 9: Archivo DEXP con elemento duplicados
Caso 10: Archivo ELIMINADOS no existe, pero puede ser creado
Caso 11: Archivo ELIMINADOS no existe y no puede ser creado
Caso 12: Archivo ELIMINADOS existe y es escribible
Caso 13: Archivo ELIMINADOS existe y no es escribible
Caso 14: Archivo ELIMINADOS vacío
Caso 15: Archivo ELIMINADOS con un elemento
Caso 16: Archivo ELIMINADOS más de un elemento
Caso 17: Archivo ELIMINADOS lleno
Caso 18: Archivo ELIMINADOS con elemento duplicados
Caso 19: Eliminar el primer elemento de DEXP
Caso 20: Eliminar el último elemento de DEXP
Caso 21: Eliminar todos los elementos de DEXP
Caso 22: Eliminar todos los elementos de DEXP menos uno
Caso 23: Eliminar un elemento de DEXP
Caso 24: Eliminar ningún elemento de DEXP

Cobertura de Instrucciones y Cobertura de Decisiones

Cobertura de Instrucciones

Es una de las técnicas más nombradas por los programadores, pero no por ello más u-
sada. Esta técnica establece que todas las instrucciones de un programa se ejecuten
al
menos una vez durante la prueba.

Esta técnica si bien es necesaria, no es en absoluto suficiente para pensar que un pro-
grama esta exento de errores. Con el segmento de código siguiente ilustraremos como
funciona la Cobertura de Instrucciones.

```
M: PROCEDURE
IF ((A>1) AND (B=0)) THEN DO;
                                X = X/A;
                                END;
IF ((A=2) OR (X>1))  THEN DO;
                                X = X +
1;
                                END;
END;
```

Cobertura de Instrucciones y Decisiones

El caso :

A=2
B=0
X=3

satisface el criterio de Cobertura de Instrucciones. Sin embargo, este criterio adolece
de los siguientes problemas:

1. La primera decisión podría haber sido codificada como un “OR” en lugar de un “AND” y el error no habría sido detectado

2. La segunda condición podría por error decir “X>0” y tampoco sería detectado.

A objeto de mejorar estos problemas se aplica la técnica de **Cobertura de Decisiones.**

Cobertura de Decisiones.

Esta técnica establece que es necesario escribir un conjunto de casos de pruebas como para que cada condición tenga, por lo menos un resultado Verdadero y uno Falso.

Los siguientes casos de pruebas satisfacen el criterio de Cobertura de Decisión:

a. A=3, B=0, X=3
b. A=2, B=1, X=1

Sin embargo, si en la segunda condición se escribe por error "X<1" el error no será detectado por ninguno de los dos casos de pruebas ya que en ambos casos el resultado de la condición es el mismo.

Como se dijo, la Cobertura de Instrucciones y Condiciones no es suficiente, pero, si necesaria.

Test de Usabilidad

Objetivos

- Mejorar el atractivo del producto
- Mejorar la amistosidad y comprensibilidad
- Detectar defectos oportunamente
- Ahorrar tiempo y recursos

Test de Usabilidad

- **Accesibilidad :** ¿Cuán fácil puedo ingresar, navegar y salir del producto?.
- **Respuesta :** ¿Satisface las necesidades del usuario haciendo lo que se necesita y cuando se necesita ?
- **Eficiencia :** ¿Realiza un mínimo de pasos ?
- **Comprensibilidad :** ¿El usuario entiende el producto, su “help” y su documentación?

Test de Usabilidad

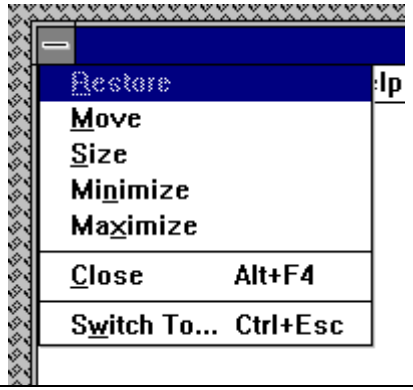
•Cómo hacer un test de Usabilidad

Clarificar los objetivos del Test

- Definir los temas en forma precisa
- Planificar las tareas y elementos de apoyo
- Preparar el plan de pruebas
- Preparar cuestionarios pre y post test
- Correr una prueba piloto y revisar el plan
- Confirmar operación del HW, video, etc.
- Realizar el test incluyendo a los desarrolladores
- Analizar los resultados
- Una forma alternativa es realizar una sesión de inspección de pantallas impresas en papel o transparencias

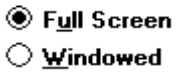
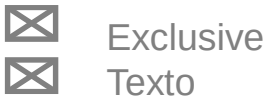
Check List de Usabilidad

Windows Compliance (extracto)

ITEM	Control	TIPO	NC	Frec.
	1.2 Para cada ventana de la aplicación			
1.	Si la ventana tiene un botón para Minimizar, Haga click en él.  Window debería desplegar un icono en pie de la pantalla. Este ícono debería corresponder con el ícono original de la aplicación.			
2.	Doble Click en el ícono debe devolver la ventana a su tamaño original.			
3.	El encabezado de una ventana debería tener el ícono de la aplicación, nombre de la aplicación y el nombre de la ventana.			
4.	Verifique que el nombre de la ventana tiene sentido.			
5.	Si la ventana tiene un menú de control, entonces use todas las opciones en gris. No deberían activarse. 			

Check List de Usabilidad

Windows Compliance (extracto)

ITEM	Control	TIPO	NC	Frec.
	1.4 Opción (Radio Buttons) 			
1.	Flechas izquierda y derecha deberían mover Selección 'ON'. Debería subir y bajar. Seleccione con Click en el mouse.			
	1.5 Check Boxes 			
1.	Click con el mouse en la caja, o en texto debería SET/UNSET la caja. SPACE debería hacerlo mismo.			
	1.6 Botones de Comando 			
1.	Si el botón de Comando conduce a otra pantalla, y si el usuario puede entrar o cambiar detalles en la otra pantalla entonces el texto en el botón debería ser seguido por tres puntos.			
2.	Todos los Botones excepto para OK y Cancel deberían tener una letra para accesarlo. Esto es identificado por una letra subrayada en el botón de texto. El botón debería ser activado presionando ALT+letra. Asegúrese de que no haya duplicación.			

Pruebas de GUI y WEB

Pruebas de GUI y WEB

El desarrollo y funcionamiento de aplicaciones GUI y WEB presenta múltiples características, las cuales agregan una complejidad adicional al proceso de pruebas.

A continuación se verá una lista, no exhaustiva, de estas características.

Para cada una de ellas es necesario contar con listas de chequeos que apoyen el diseño de casos de pruebas.

Algunos tipos de pruebas a considerar

- **Diferencias de Browsers**
  - Internet Explorer, Netscape
  - Distintas versiones para diversos sistemas operativos
  - Múltiples idiomas
- **Validación de lenguajes**
  - El Browser interpreta el código HTML y despliega la página en el monitor
  - Es recomendable validar que el lenguaje sea estándar
- **Imágenes**
  - Diagramas, mapas, fotografías
  - Verificar tiempos de carga, formato, calidad de despliegue
  - Links asociados a imágenes sensitivas
- **Fonts**
  - Tipos, Colores, tamaños,...

Pruebas de GUI y WEB

ITEM	Control	TIPO	NC	Frec.
	1.4 Validación de FONTS			
	El Font es proporcional o fijo según se requiera			
1.	El Font primario es estándar			
2.	Sólo se usan tamaños estándares (small, médium Large) y no tamaños con número de puntos específicos			
3.	No se usan más de 3 tipos de fonts			
4.	Fonts de símbolos son usados sólo cuando son absolutamente necesarios			
5.	Los colores por defecto del Browser se respetan			

Pruebas de GUI y WEB

Algunos tipos de pruebas a considerar

- **Tablas**
  - (Dimensiones, textos, líneas, imágenes incrustadas,...)
- **HW del cliente**
  - (PC, Laptop, celular, mm, speed, ...)
- **SW del cliente**
  - (Windows, Windows NT, Mac,...)
- **Aplicaciones de terceros (Plug in)**
  - (Audio, RealPlayer, Acrobat, Webcelerator...)
- **Links**
  - (Internos y externos)
- **BookMarks y Favoritos**

Pruebas de GUI y WEB

ITEM	Control	TIPO	NC	Frec.
	1.3 Validación de imágenes			
1.	Está imagen agrega valor a la página			
2.	El tipo de imagen es apropiado (Jpg para fotos; Gif para botones)			
3.	El tamaño de la imagen es apropiado para el tamaño del monitor en que será desplegada			
4.	El tamaño físico de esta imagen comprimida es el más pequeño posible sin comprometer la calidad de la imagen			
5.	Hay un ALT tag para esta imagen			
6.	Los tags WIDTH y HEIGTH están expresados en términos de porcentaje y no en valores absolutos			
7.	La imagen tiene los derechos legales para ser usada			

Algunos tipos de pruebas a considerar

- **Frames**
  - Permiten definir una o más secciones “Scrollables” (uso de barras)
  - Los frames pueden ser transmitidos en forma indpendiente unos de otros
- **Impresión de páginas**
  - Verificar tamaño, fonts, colores, ...
- **Máquina de Búsqueda interna** (Ver checklist en pág. sgte)
  - Permite realizar búsqueda de textos y/o documentos contenidos en la página
- **Formularios**
  - (Validación contenido, largo, tipo de datos)
- **Residencia de validación**
  - (Generalmente son mínimas en cliente y mayores en el servidor)

Checklist para Máquinas de Búsqueda

ITEM	Control	TIPO	NC	Frec.
	1.1 Para Cada Búsqueda			
1.	El resultado de la búsqueda es devuelto dentro de 5 segundos (excluyendo el tiempo de transmisión internet)			
2.	El conjunto de resultados está ordenado			
3.	La búsqueda funciona si se ingresan datos comunes (Ej: “a”)			
4.	La búsqueda funciona si se ingresan datos poco comunes (Ej: “fdrer^=*”)			
5.	La búsqueda ignora el código fuente usado para construir la página y sólo busca en el contenido de los documentos definidos.			
6.	La búsqueda es sensible a palabras tales como “secreto”.			
7.	La búsqueda funciona con el máximo de caracteres (y también con el máximo más 1)			
8.	La búsqueda reconoce operadores lógicos			
9.	La búsqueda reconoce wildcards			

Algunos tipos de pruebas a considerar

- **Páginas dinámicas**
  - Manipuladas por script en el cliente
  - Permiten cambiar el aspecto de la página durante la sesión
- **Campos Pop up´s**
  - Manejados a través de scripts transmitidos con la página madre
  - Pueden manejar scroll, resizable, back, forward,..
  - Se recomienda mantener el estandar de la aplicación madre (GUI o Browser)
- **Flujo de Contenido**
  - Pull: El usuario solicita la información
  - Push: El servidor entrega la información sin solicitud del usuario. Requiere estar conectado

Algunos tipos de pruebas a considerar

- **Paso de datos entre un Usuario WEB y una aplicación**
  - Cuando un usuario ingresa datos en una página WEB, estos datos son pasados por la página WEB a una aplicación que los procesa.
  - Este traspaso normalmente requiere de un software (Script). Este software puede estar escrito en un lenguaje como C, por ejemplo.

- **Acceso de datos a una Base de Datos**
  - La página WEB puede accesar directamente la BD vía SQL en el Script
  - La página WEB puede accesar la BD via Sript a una aplicación y ésta última tiene el SQL necesario para accesar la BD

Algunos tipos de pruebas a considerar

- **Cookies**
  - Las cookies son archivos de datos (no son software) que el servidor Web envía junto con la página
  - Cuando el usuario pide otra página al mismo servidor la Cookie es devuelta al servidor.
  - El servidor Web usa estos datos para rastrear el acceso del usuario a ciertas páginas, optimizar los tiempos de respuesta, etc.
  - Un servidor Web no debería retornar cookies de otro servidor

CAST
Computer Aided Software Testing

CAST
Computer Aided Software Testing

Técnica de Causa-Efecto

- **Verifica la funcionalidad**
  1. Completitud
  2. Consistencia
  3. Ambigüedad

- **Diseño de Casos de Prueba**
  1. Verificar que el software cumple con los requerimientos

DISEÑO DE CASOS DE PRUEBA

•Lo que deseamos cuando planificamos nuestros test :

Una especificación completa, no ambigua, consistente.

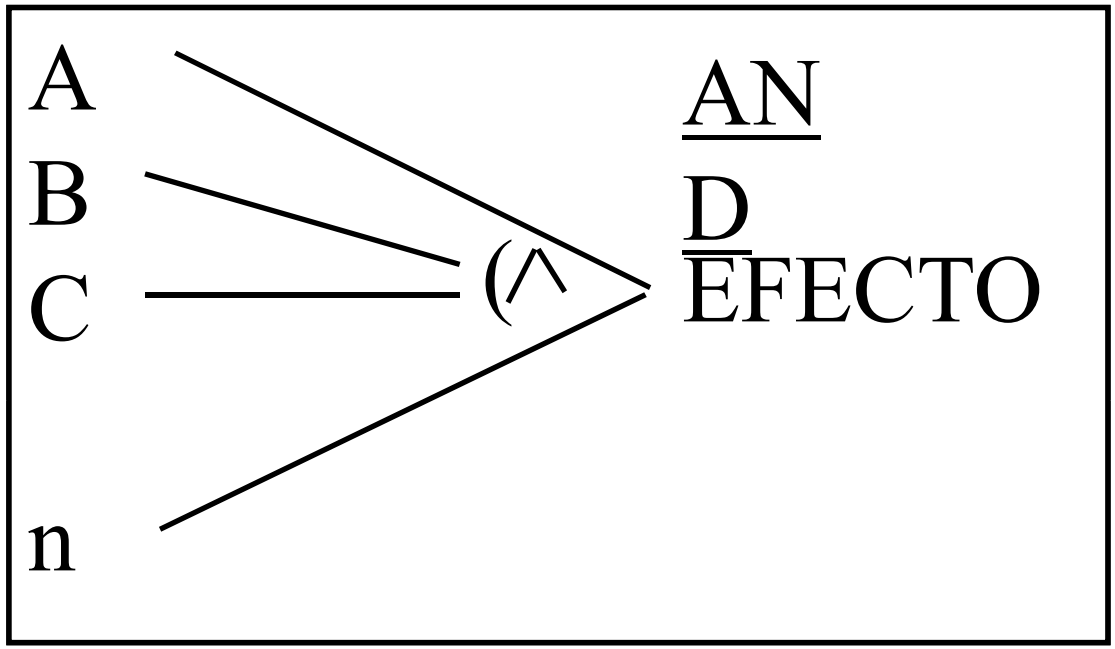
• Lo que generalmente tenemos :

Un resumen de requerimientos
Alguna documentación del diseño
Manuales antiguos
Nuestro propio conocimiento
Rumores

Análisis de Casos Posibles

Nº Entradas	Nº Variacion	Nº Comb.	Nº Fallas
n	n+1	n	3^n-1
2	3	4	8
3	4	8	26
4	5	16	80
5	6	32	242
6	7	64	

VARIACIONES FUNCIONALES



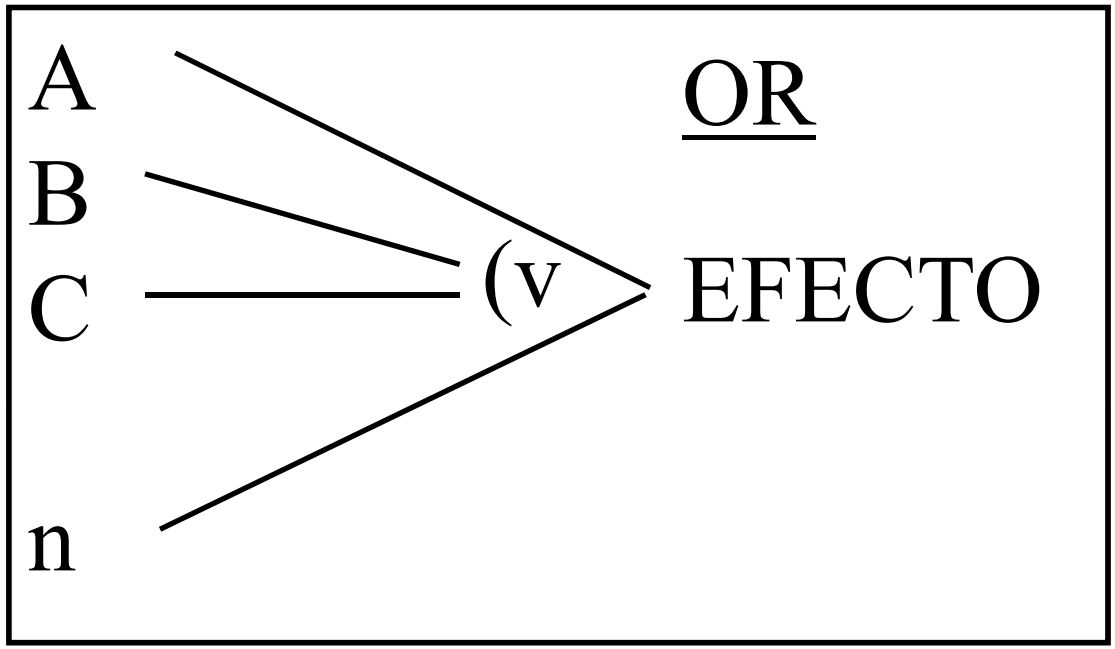
1. A(f),B(v),C(v),...,n(v) ent. Efecto ausente
2. A(v),B(f),C(v),...,n(v) ent. Efecto ausente
3. A(v),B(v),C(f),...,n(v) ent. Efecto ausente

. . .

n. A(v),B(v),C(v),...,n(f) ent. Efecto ausente

n+1A(v),B(v),C(v),...,n(v) ent. Efecto

VARIACIONES FUNCIONALES



1. A(v),B(f),C(f),...,n(f) entonces Efecto
2. A(f),B(v),C(f),...,n(f) entonces Efecto
3. A(f),B(f),C(v),...,n(f) entonces Efecto

. . .

n. A(f),B(f),C(f),...,n(v) entonces Efecto

n+1A(f),B(f),C(f),...,n(f) ent. Efecto ausente

Especificación en Causa-Efecto

Ejemplo

Se debe desarrollar una rutina de cajero automático, que actúe de acuerdo a lo siguiente : Si la tarjeta del cliente es válida y el número de identificación ingresado por el cliente es válido y el saldo del cliente en la cuenta es mayor que $20.000,00 se debe aceptar la transacción.
En caso contrario se debe rechazar.

Especificación en Causa-Efecto

Versión en Causa-Efecto

```
TITLE 'Cajero'.

NODES
Tar_val      = 'Tarjeta es válida'.
Num_val   = 'Número Id. válido'.
Saldo>20   = 'Saldo en cta. >
$20.000,00'.
Aceptar_Tr= 'Aceptar transacción'.
Rechaz_Tr = 'Rechazar transacción'.

RELATIONS
Aceptar_Tr :- Tar_val and Num_val
                  and Saldo>20.
Rechaz_Tr  :- Not Aceptar_Tr.
```

Ejercicio Nº 5

Este módulo lista el archivo de prestaciones de la clínica MASVIDA. Por cada registro del archivo genera un registro en el listado, de acuerdo a las siguientes reglas de proceso:

Se recibe un código por pantalla. Si éste es L1 se emite el listado llamado L1. Si el código tiene cualquier otro valor se cancela el proceso.

Por cada paciente moroso se debe generar un registro en el Tabla MOROSOS de la base de datos de PACIENTES. Un paciente está moroso si la fecha de vencimiento del registro es < Fecha de Hoy del computador. Según la cantidad de días corridos entre Esas dos fechas los pacientes MOROSOS se clasifican en los siguientes casos:

Moroso < 30 días	tramo 1
Moroso >= 30 días < 60 días	tramo 2
Moroso >= 60 días	tramo 3

Dependiendo del caso , el registro en la Tabla MOROSOS llevará el campo TIPO_Moroso = 30, 60 o 90.

CAST
Computer Aided Software Testing

Para los pacientes morosos del tramo 1 o tramo 2 debe imprimirse Carta Aviso. Para los morosos que estén en el tramo 3 debe imprimirse Carta Aviso y, además, el anexo Suspensión de Prestaciones.

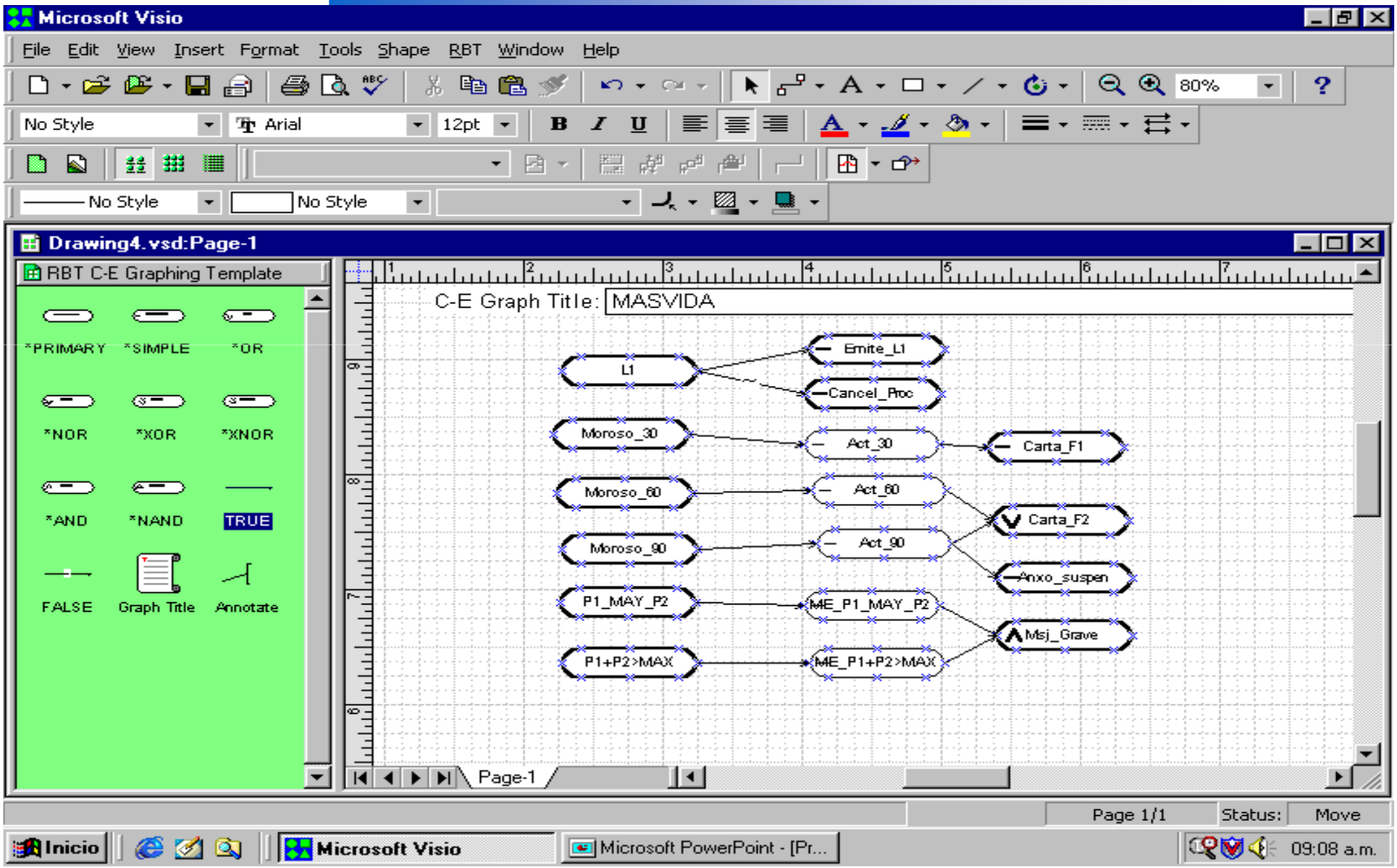
Los morosos que estén en los tramos 2 o 3, deben tener carta formato 2. Los demás carta formato 1.

Cada paciente puede tener dos tipos de prestaciones : tipo 1 y tipo 2. Estas prestaciones deben cumplir con lo siguiente:

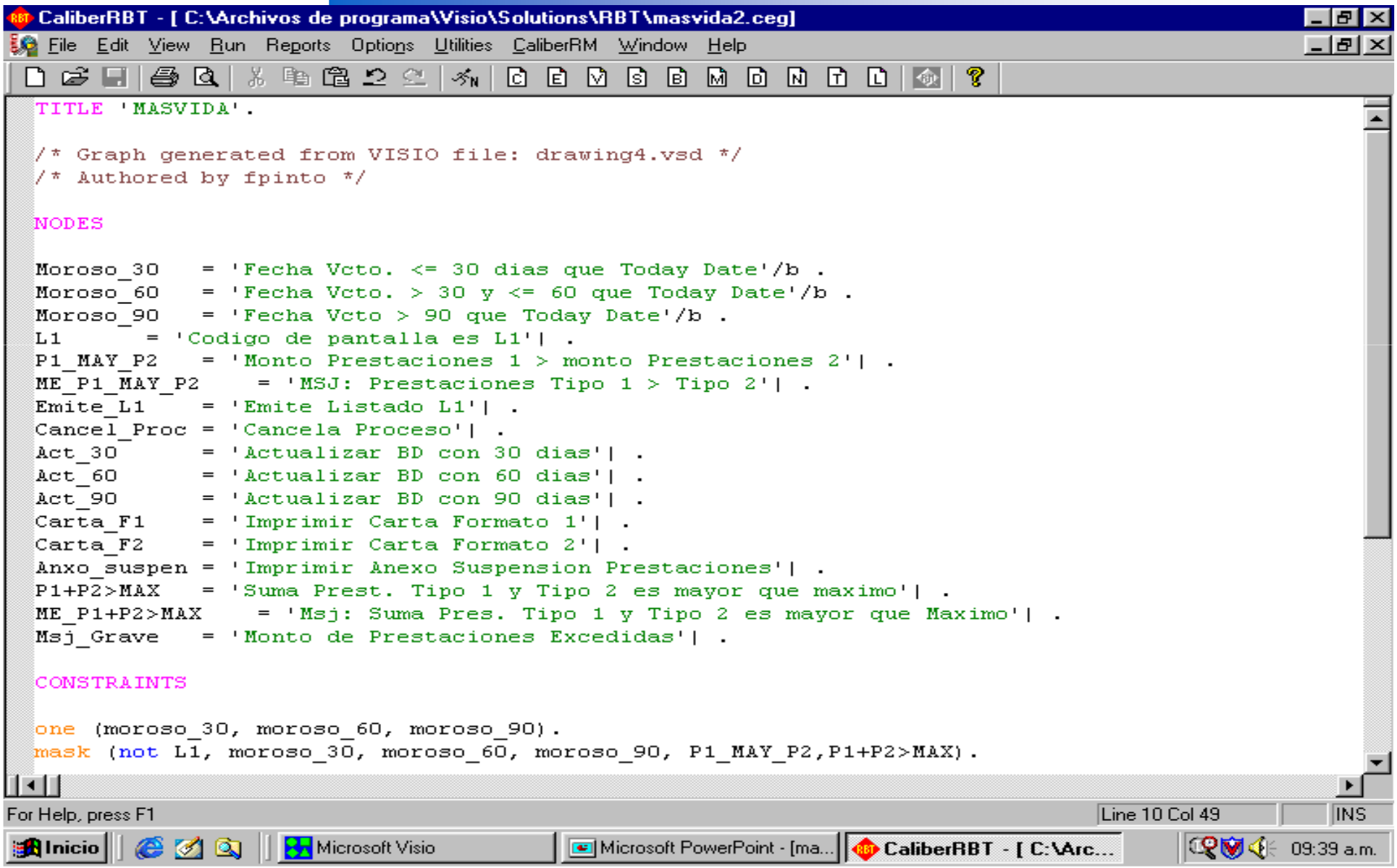
El campo Monto Prestaciones Tipo 1 no debe ser mayor que el campo Monto Prestaciones Tipo 2. La suma de las prestaciones tipo 1 y tipo 2 de cada paciente no debe ser superior al campo Monto Máximo Autorizado.

Cualquiera de estas situaciones que ocurra debe ser informada junto con el registro. En el caso que ocurra el caso 1 y el caso 2, se debe imprimir el mensaje : ** Error Grave **.

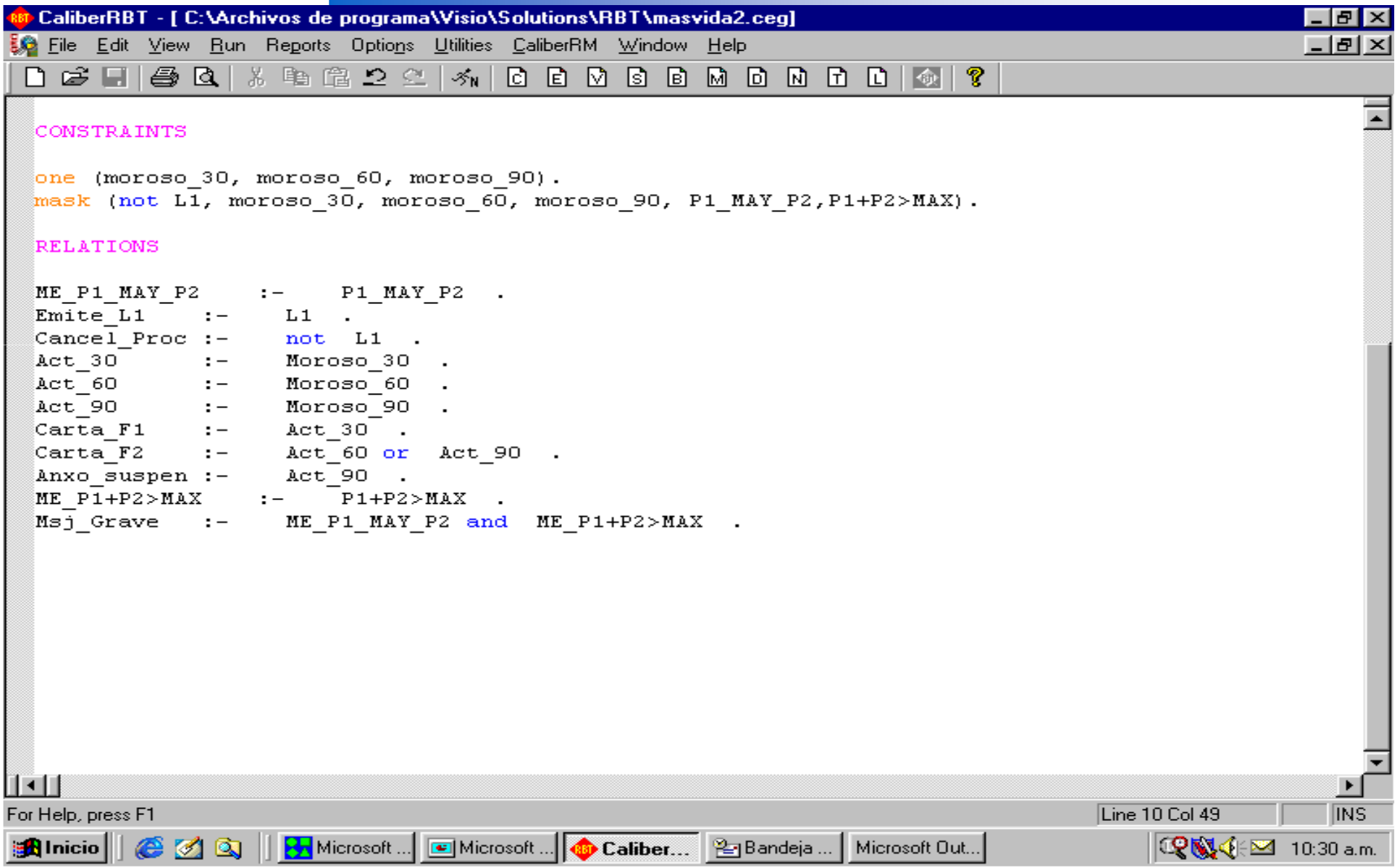
CAST
Computer Aided Software Testing



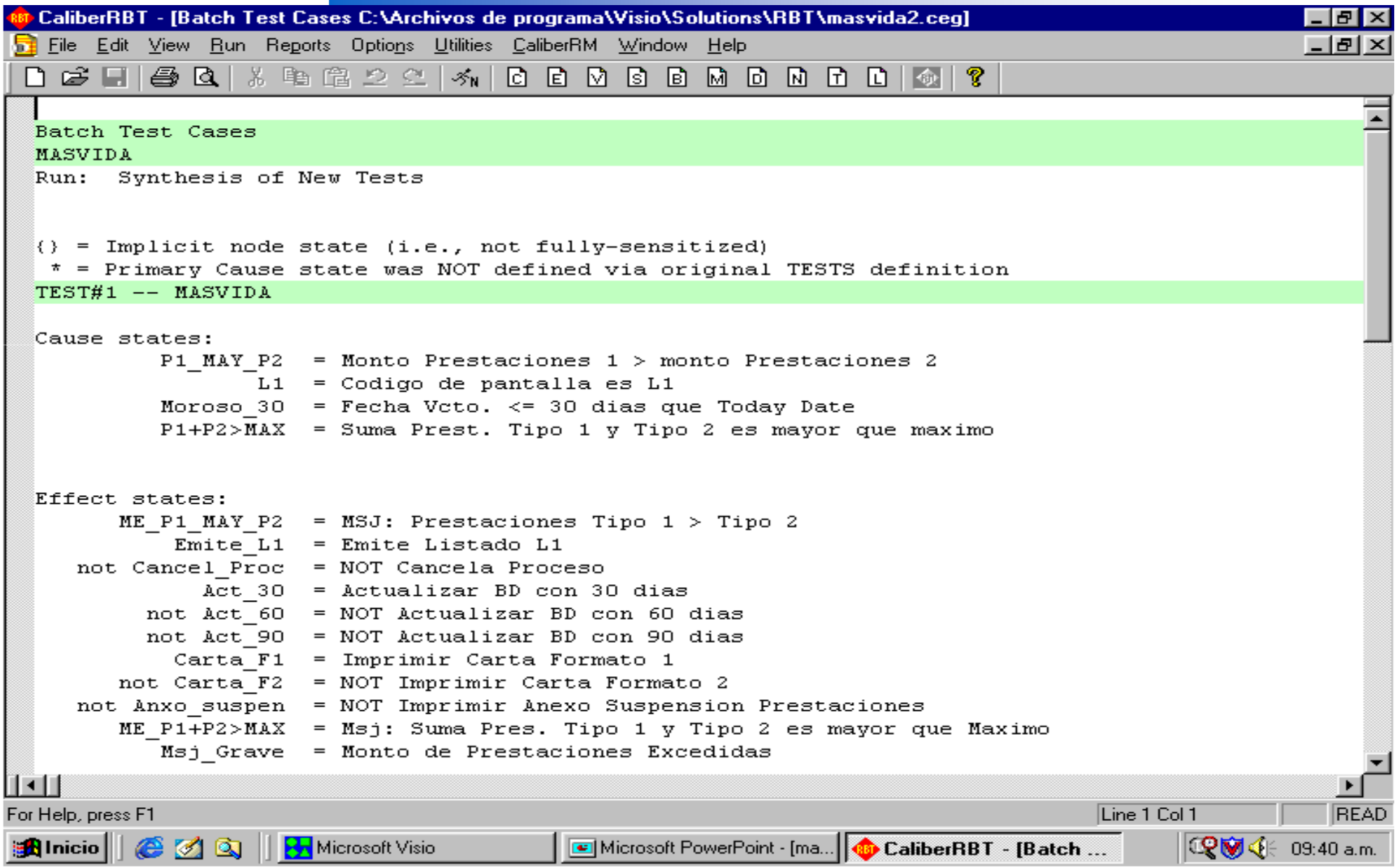
CAST
Computer Aided Software Testing



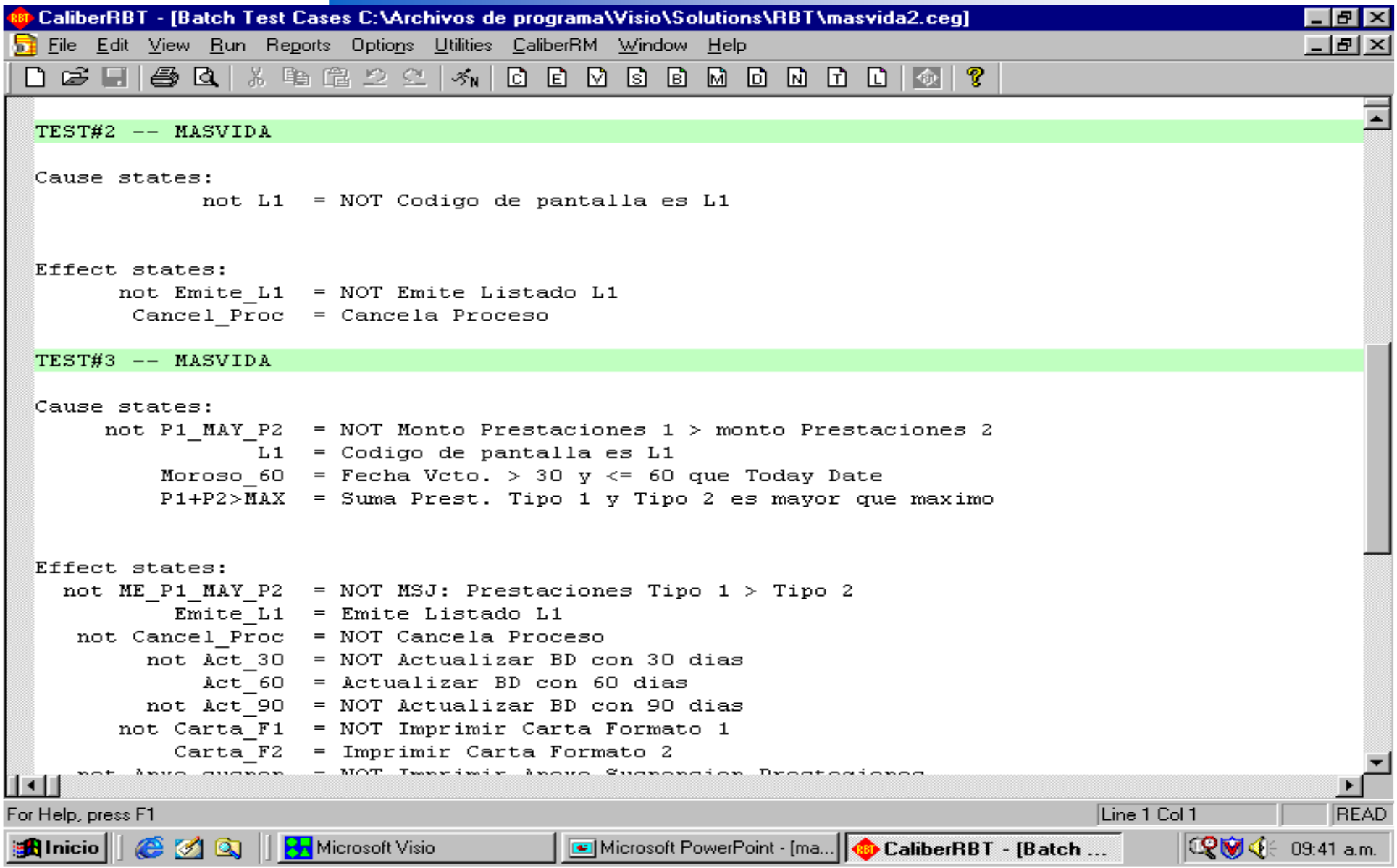
CAST
Computer Aided Software Testing



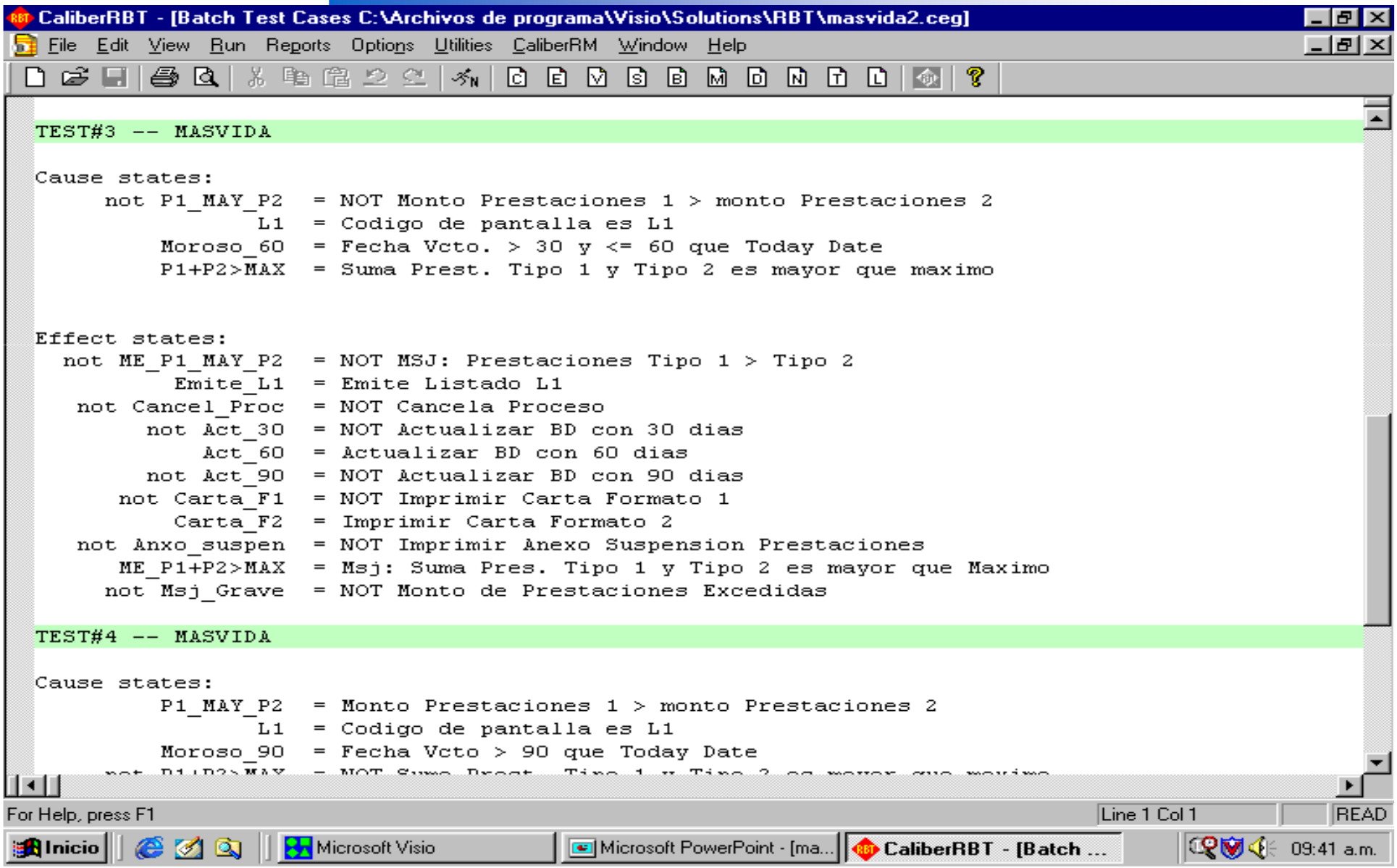
CAST
Computer Aided Software Testing



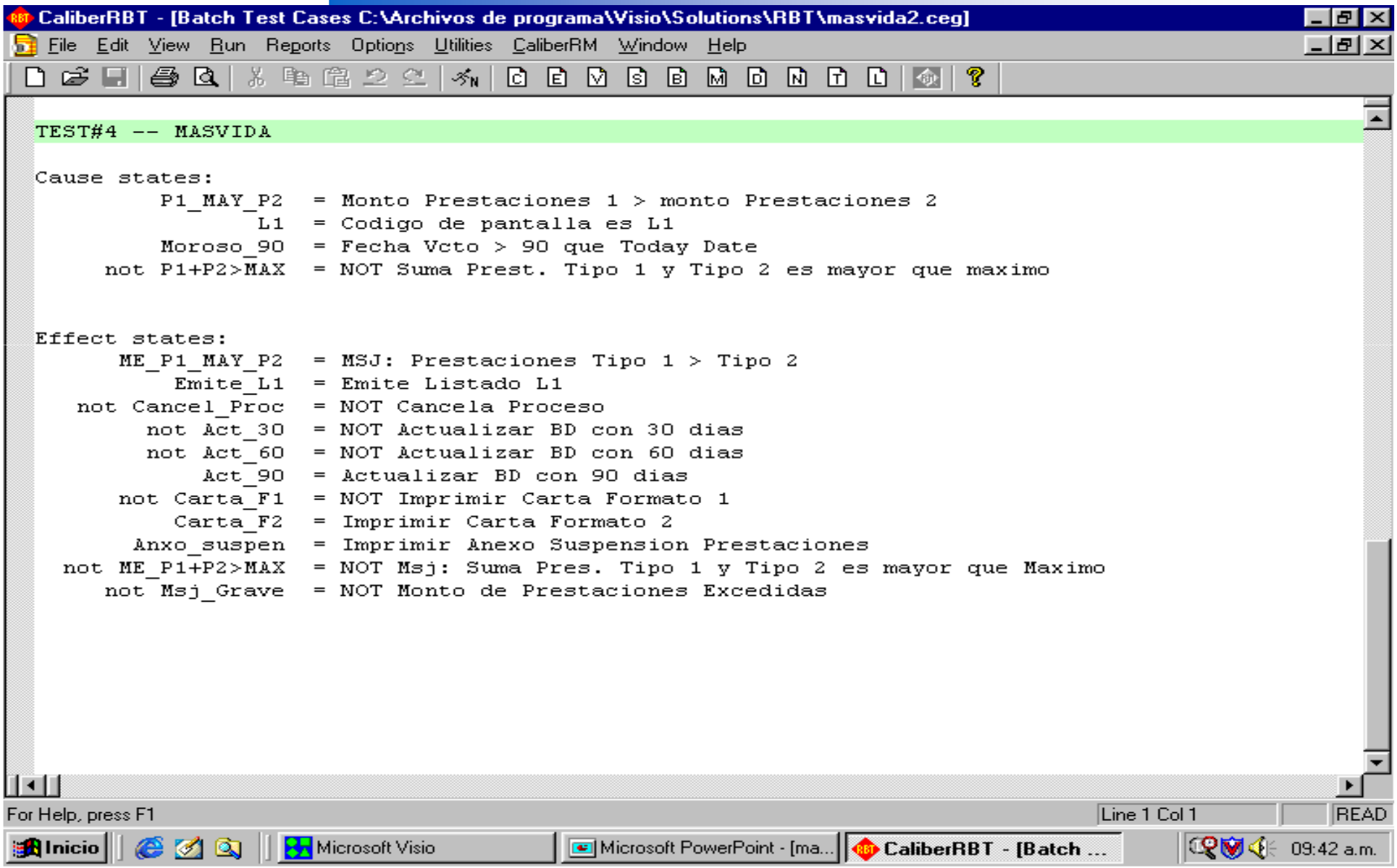
CAST
Computer Aided Software Testing



CAST
Computer Aided Software Testing



CAST
Computer Aided Software Testing



CAST
Computer Aided Software Testing

CAPTURE/PLAYBACK

- Técnica muy usada para automatizar el test de regresión
- La herramienta “captura” los inputs y outputs del sistema
- La herramienta probadora reejecuta con los datos originales y compara resultados
- Requiere una versión previa del software
- No genera los casos de prueba

CAPTURE/PLAYBACK

Algunas características de las herramientas actuales

- Permiten identificar cambios menores sin que sean considerados errores (Ej. Ubicación de información, tamaño de botones, cambio de colores, etc.)
- Permiten especificar rangos de validez de una variable
- Permiten definir puntos de control para controlar valores de salidas específicos (Ej. Valor esperado en un campo)
- Identificación de bloques de código permiten simplificar la tarea de codificación al intervenir un Test Script

Prueba de Sistemas

Prueba de Sistemas

El objetivo de la Prueba de Sistemas es probar el sistema como un todo. A esta altura del proceso ya se han probado los programas en forma individual, integrada y su funcionalidad. La Prueba de Sistemas cubre aquellas partes de la especificación que no han sido probadas en las pruebas anteriores, relativas a procesos, espacio, tiempos de respuesta, etc.

No existen técnicas formales para generar casos de pruebas para realizar la Prueba de Sistemas, por lo que se opta por identificar un conjunto de categorías de pruebas que deben realizarse.

No todos los sistemas deben ser probados en todas estas categorías, pero, si para todos los sistemas debe verificarse cuales de ellas son procedentes.

Realización de la Prueba de Sistemas

En la realización de la pruebas de sistemas es conveniente tener en cuenta los siguientes criterios:

1. La Prueba de Sistema debe realizarse con un conocimiento profundo de como el sistema funcionará en su medio
2. La prueba debe ser guiada por un representante del usuario, y apoyado por un analista y un diseñador
3. El enfoque debe ser tratar de demostrar que el sistema no cumple con lo que se especificó
4. Debe ser realizada por personas independientes de aquellos que desarrollaron el sistema
5. La unidad de desarrollo debe apoyar en la generación de ambientes y otros aspectos técnicos a los probadores

Realización de la Prueba de Sistemas

- **Resistencia:** El objetivo es probar el comportamiento del sistema ante volúmenes máximos de datos, conexiones simultáneas, etc.

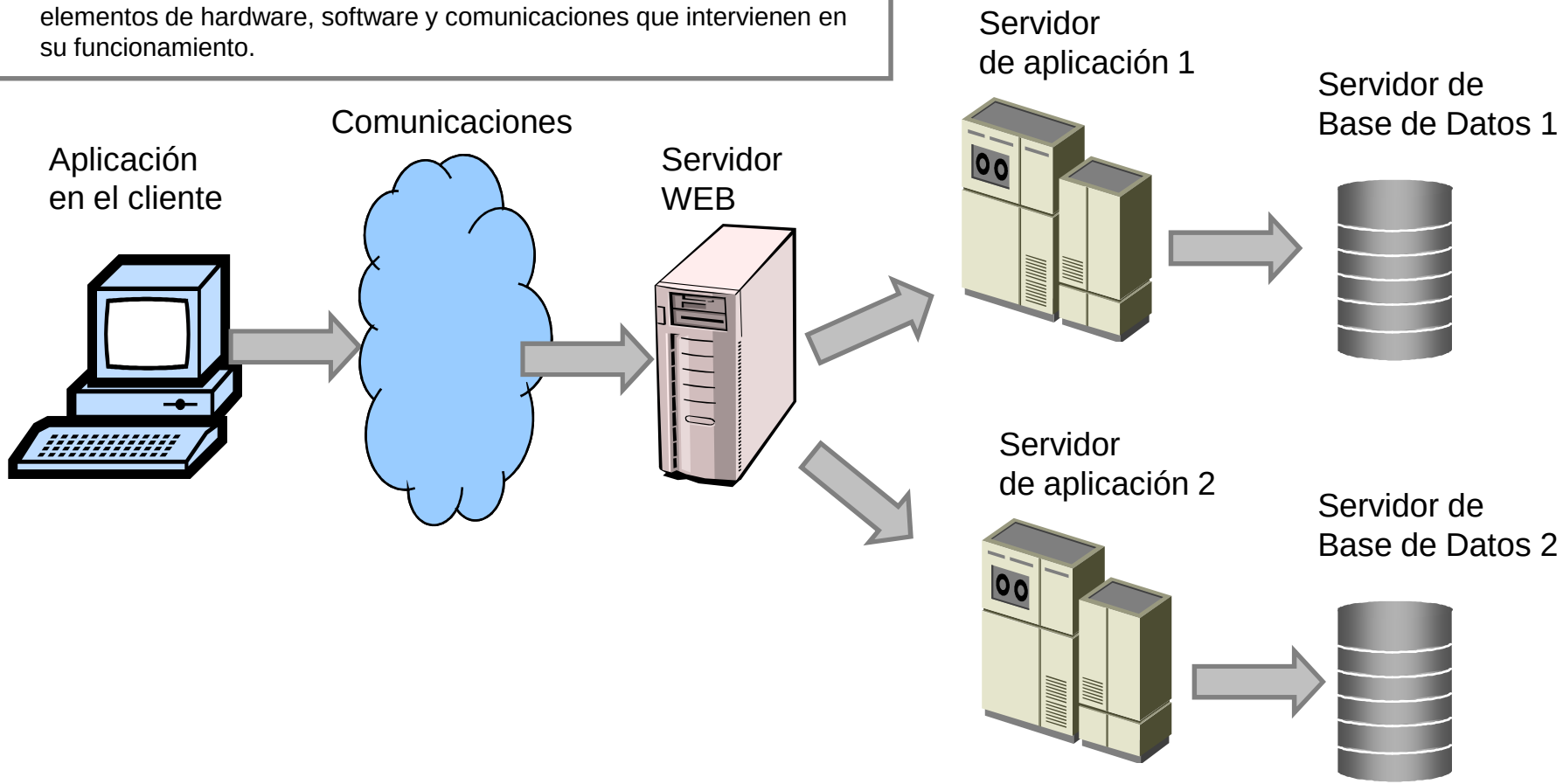
- **Rendimiento:** El objetivo es probar el tiempo de respuesta del sistema ante volúmenes medios de datos, conexiones simultáneas, etc.

- **Seguridad:** El objetivo es probar la seguridad del sistema ante accesos no autorizados al software y los datos.

- **Recuperación:** El objetivo es probar la capacidad de recuperación ante fallas que tiene la aplicación.

Realización de la Prueba de Sistemas

El incremento explosivo que las Tecnologías de Información tienen en la actualidad, hace que cada día sea más complejo predecir el comportamiento de una aplicación. Consideremos los múltiples elementos de hardware, software y comunicaciones que intervienen en su funcionamiento.



Realización de la Prueba de Sistemas

Es necesario planificar y desarrollar estrategias de testing específicamente orientadas a verificar el comportamiento de las aplicaciones una vez que estas sean puestas en Producción.

En la actualidad estas pruebas están orientándose al uso de herramientas de software que ayudan en la realización de ellas. Algunas funciones que son abordadas son las siguientes:

- Simulación de Ambientes de Producción (múltiples componentes de hw y sw)
- Procesar grandes cantidades de transacciones
- Utilizar recursos de hardware disponibles
- Simular transacciones, tiempo de espera de usuario,....
- Repetir pruebas

Estas pruebas permiten:

- Determinar cuellos de botella de hardware y software
- Determinar los % de incidencia de cada componente
- Establecer acuerdos de Nivel de Servicio (SLA)
- Presupuestar crecimiento y costos

Realización de la Prueba de Sistemas

Algunas recomendaciones en relación a la prueba de sistemas

La organización debe considerar los siguientes aspectos:

- Adquisición o arriendo de herramientas
- Disponibilidad de personal capacitado en el uso de herramientas
- Periodicidad de este tipo de pruebas
- Tamaño de la aplicación a probar
- Arquitectura utilizada
- Criticidad de la aplicación para el negocio
- Riesgos
- Costos

Tipos de Pruebas según Caper Jones

Tipos de Pruebas

Los 18 tipos de Pruebas según Capers Jones

Las pruebas según Capers Jones se dividen en tres grandes niveles y dentro de ellas se verá cada uno de los tipos de pruebas que se considera, aquí se entrega un desglose por cada nivel y dentro de él los tipos de pruebas a considerar, al final se adjunta un resumen en donde se muestra los tipos de pruebas más usados, toda esta información corresponde a empresas de USA.

Las Pruebas Generales de Software

1.- Pruebas de subrutina:

Corresponde a la prueba de una subrutina específica, como por ejemplo Cálculo del DV del RUT

Es la forma de nivel más bajo de probar. Una subrutina es una pequeña colección de código que puede constituir menos de 10 declaraciones o quizás un décimo de un punto de función.

Tipos de Pruebas

2.- Pruebas de unidad:

Corresponde a la prueba que ejecuta el programador cuando termina su programa.
Las pruebas de unidades son la ejecución de un módulo completo o un programa.
La Inspección de Códigos es una Técnica de detección de errores, que debe (o puede) aplicarse una vez que el programa es liberados por el autor y antes de efectuar las pruebas del SW.

3.- Pruebas de nuevas funcionalidades:

Corresponde a la prueba que ejecuta el área de Testing para determinar que las funcionalidades entregadas se cumplan en el contexto del requerimiento de usuario.
A menudo es combinado con pruebas de regresión, y ambas formas comúnmente son encontradas cuando programas existentes están siendo puestos al día (actualizados) o modificados. Como el nombre indica, la prueba de nuevas funcionalidades es apuntada a la validez de las nuevas características (funciones) que están siendo añadidos a un paquete de programas de software.

Tipos de Pruebas

4.-Las pruebas de regresión

Corresponde a la prueba que permite volver a un punto determinado por una falla en el sistema, por cambios o por nuevas funcionalidades.
La palabra "regresión" quiere decir volver atrás y en el contexto de probar significa verificar que no se haya producido daño de una característica existente al añadir una nueva característica. La regresión también prueba o comprueba que errores previos conocidos sin querer se han quedado en el software después de que ellos deberían haber sido quitados. La prueba de regresión puede ser realizada por desarrolladores o personal especializado en pruebas.

5.- La prueba de integración

Corresponde a la prueba que permite revisar la integración de diferentes piezas, módulos o sistemas.
Como el nombre lo indica, prueba varios módulos o programas que tiene que trabajar juntos para constituir un paquete de software integrado. Las pruebas de integración pueden cubrir el trabajo de docenas o aún cientos de programadores. Esto también trata con los números bastante grandes de casos de prueba. Estas pruebas de integración pueden realizarse por sub-conjunto de módulos que son construidos por ejemplo semanalmente.

Tipos de Pruebas

6.- Las pruebas de sistema

Corresponde a la prueba que permite probar el sistema en su conjunto. Por ejemplo funcionalidades, integración, rendimiento.

Son por lo general la última forma de pruebas internas antes de que los clientes estén implicados con la prueba sobre el terreno (pruebas de Beta). Para sistemas grandes, una prueba de sistema formal puede tomar muchos meses y puede implicar grandes equipos de personal de prueba. También, el grupo entero de programadores de desarrollo puede ser necesario para remover los errores que son encontrados durante esta etapa crítica de prueba.

Tipos de Pruebas

Las Pruebas Especializadas de Software

1.- Stress o pruebas de capacidad

Corresponde a la prueba que permite determinar como responde el sistema ante diferentes necesidades.
Son una forma especializada de probar apuntado a la evaluación de la capacidad de funcionar acercándose a las fronteras de sus capacidades en términos del volumen de información usada. Por ejemplo, las pruebas de capacidad del procesador de texto se solían crear un libro (Microsoft Word para la Versión 7 de Ventanas) que podría implicar pruebas contra los documentos individuales grandes de quizás 200 a 300 páginas para juzgar los límites superiores que pueden ser manejados antes de que MS Word se haga incómodo o el almacenaje sea excedido. Esto también podría implicar tratar otros documentos aún más grandes, decir 2000 páginas, segmentadas en documentos de maestro y varias secciones. Para un uso de base de datos, las pruebas de capacidad podrían implicar la carga de la base de datos con 10000 o 100000 o 1000000 de registros para juzgar como esto funciona cuando el sistema está totalmente poblado con la información.

Tipos de Pruebas

2.- Las pruebas de funcionamiento

Corresponde a la prueba que permite ver el funcionamiento de una pieza en aspectos específicos y relevantes de la aplicación

Son una forma especializada de probar apuntado a la valoración si realmente se puede encontrar los objetivos de funcionamiento dispuestos para ello. Ya que mucho funcionamiento de usos es sólo una cuestión menor, pero para algunas clases de usos es crítico. Por ejemplo, los sistemas de armas, sistemas de control de vuelo de avión, sistemas de inyección de combustible, métodos de acceso, y sistemas de conmutación de teléfono deben encontrar objetivos de funcionamiento rigurosos o los dispositivos el software controla no puede trabajar.

3.- Pruebas de protección de virus:

Corresponde a la prueba que permite determinar si las piezas contienen algún virus o pueden ser “hackeadas” fácilmente.

Rápidamente se mueve de una prueba especializada a una general, aunque todavía se aplica sobre menos de la mitad de los proyectos de nuestro cliente. La introducción de virus de software por hackeres malévolos han sido los fenómenos sociológicos más interesantes en el mundo de software. La cantidad de miles de virus y más que está siendo creada diariamente (a diario) debería hacernos pensar en este tipo de pruebas.

Tipos de Pruebas

4.- Las pruebas de Seguridad

Corresponde a la prueba que permite determinar que el SW (Programas, archivos, líneas comunicación..) está protegido contra intervenciones externas no autorizadas.
Son las más comunes y las más sofisticadas para el software militar, seguido del software que trata con la información muy confidencial como registros bancarios, expediente médico, registros fiscales, y otros por el estilo

5.- Las pruebas de plataforma

Corresponde a la prueba que permite determinar que el SW se ejecuta sin problemas en las plataformas que se definió para que funcionara.
Son una forma especializada de probar usado en empresas cuyo software funciona sobre plataformas de hardware diferentes bajo sistemas diferentes de operaciones. Mucho software comercial es desarrollado para Windows 95, Windows NT, OS/2, UNIX, y a veces para otras plataformas también, con las mismas funcionalidades en cada plataforma.

6.- Probando el año 2000

Corresponde a la prueba que se ejecutó para el cambio de siglo.
El problema del año 2000 es que se usan sólo dos dígitos para fechas (año),

como 96 hacia 1996 y el problema surge cuando se pasa de 1999 a 2000 porque los 00, si se maneja en dos dígitos interrumpirán cálculos. Estas pruebas puede que aún tengan sentido en sistemas construidos antes del año 2000 o que procesen datos anteriores al año 2000.

7.- Pruebas independientes

Corresponde a la prueba que ejecuta un ente externo y puede ser cualquiera de las ya definidas.
Pruebas independientes, como el nombre implica, son realizadas por una empresa separada o al menos una organización separada de la que construyó el sistema. Tanto la caja blanca como las formas de caja negra de pruebas independientes son usadas.

Tipos de Pruebas

Las Pruebas que involucran usuarios o clientes

1.- Las pruebas de Usabilidad

Corresponde a la prueba que permite ver amistosidad del sistema con el cliente y así este no tenga un problema mayor.
Son una forma especializada de pruebas a veces realizadas en laboratorios. Las pruebas de usabilidad involucran a los clientes reales que utilizan el software y a veces instrumentadas en condiciones tales que sus acciones puedan ser observadas. Las pruebas de usabilidad son comunes para el software comercial producido por empresas grandes como la IBM y Microsoft. Las pruebas de Usabilidad pueden ocurrir con cualquier clase de software pese a todo. Las pruebas de usabilidad son una forma de caja negra de pruebas y por lo general ocurren casi al mismo tiempo que la prueba de sistema. A veces las pruebas de usabilidad y pruebas de Beta son simultáneas (concurrentes), pero es más común que las pruebas de usabilidad precedan las pruebas Beta.

Tipos de Pruebas

2.- La prueba de campo (Beta)

Corresponde a la prueba que ejecutan los clientes, es lo que conocemos como Marcha Blanca.
Es una técnica de pruebas común para el software comercial. La palabra " Beta " es usada pensando en una prueba externa que implica a clientes. El test Beta generalmente se realiza después del test de sistema. Microsoft se ha hecho famoso realizando testing Beta con más de 10000 usuarios participando. (Marcha Blanca)

3.- Las pruebas de laboratorio

Corresponde a la prueba que se ejecutan en laboratorio como su nombre lo indica, se prueba hardware y software juntos (celular).
Son una forma especial de probar encontrándose principalmente en los productos híbridos que consisten en complejos dispositivos físicos que son controlados por el software, como sistemas de conmutación de teléfono, sistemas de armas, e instrumentos médicos. Es obvio que la prueba sobre el terreno convencional o las pruebas de Beta de algo como un misil de crucero ó el alambrado especial eléctrico, no es posible. Por lo tanto, las compañías que construyen estos sistemas tienen sus laboratorios donde realizan las pruebas necesarias antes de que los sistemas pasen a producción.

Tipos de Pruebas

4.- Las pruebas de aceptación de cliente

Corresponde a la prueba que ejecuta el cliente para dar por aceptado un producto.

Comúnmente son efectuadas para el software de contrato y a menudo encontradas para sistemas de información administrativa, software de sistemas, y software militar. La única forma de software donde las pruebas de aceptación son raras o no ocurre es él de software comercial de grandes volúmenes. Incluso aquí, algunos vendedores y tiendas al menudeo proporcionan una garantía de vuelta (reembolso) que permite una forma de pruebas de la aceptación. Como los clientes realizan las pruebas de aceptación varía bastante, pero por lo general las pruebas de aceptación son una forma de caja negra de pruebas.

5.-Las pruebas estadísticas del Clean-Room

Corresponde a la prueba que es encontrada sólo en el contexto de métodos de desarrollo de clean-room.

Los casos de prueba están basados en las aserciones estadísticas de modelo de uso. Las pruebas de clean-room son una forma de caja negra de pruebas y siempre son realizadas por especialistas o el personal de control de calidad más bien que por los propios desarrolladores.

Tipos de Pruebas

Estadísticas de los tipos de Pruebas

De acuerdo a la información recopilada por el autor, la mayoría de proyectos de software en los Estados Unidos (el 70 %) usa seis tipos de pruebas y el modelo más común observado de pruebas incluye el siguiente

- Prueba de Subrutina
- Pruebas Unitarias
- Pruebas de Nueva Funcional
- Pruebas de Regresión
- Pruebas de Integración
- Pruebas de Sistema

Estos seis tipos de pruebas son muy comunes sobre los sistemas de 1000 puntos de función o más grandes. Estos seis también resultan ser las formas de las pruebas que se tratan en la mayoría de las categorías de errores y publicaciones.

Tipos de Pruebas

Debajo de los 1000 puntos de función y sobre todo debajo de los 100 puntos de función, sólo tres tipos de pruebas son encontradas, asumiendo el proyecto en cuestión es nuevo y no una mejora.

Prueba de Subrutina
Pruebas Unitarias
Prueba de Nueva Funcional

Otros tipos de pruebas que son menos comunes son más especializados, como pruebas de funcionamiento o pruebas de capacidad, y tratan con una banda estrecha de problemas de los cuales no todos los sistemas están preocupados.

Tipos de Pruebas

Número de Tipos de Pruebas	% de Proyectos que la Utilizan
1 Tipo de Prueba	2%
2 Tipos de Pruebas	8%
3 Tipos de Pruebas	12%
4 Tipos de Pruebas	14%
5 Tipos de Pruebas	16%
6 Tipos de Pruebas	18%
7 Tipos de Pruebas	5%
8 Tipos de Pruebas	5%
9 Tipos de Pruebas	7%
10 Tipos de Pruebas	5%
11 Tipos de Pruebas	3%
12 Tipos de Pruebas	1%
13 Tipos de Pruebas	1%
14 Tipos de Pruebas	1%
15 Tipos de Pruebas	1%
16 Tipos de Pruebas	1%
17 Tipos de Pruebas	0%
18 Tipos de Pruebas	0%

Criterios de Término de las Pruebas

Criterio de Término de las Pruebas

¿Cuándo terminar de probar ?

Normalmente las pruebas terminan cuando se termina el tiempo planificado para probar. Es más, en muchas ocasiones este tiempo se ve reducido puesto que al estar las pruebas al final del desarrollo, esta fase debe absorber el impacto de todos los atrasos producidos en las fases anteriores.

Otro criterio comúnmente usado para detener las pruebas es cuando todos los casos de pruebas confeccionados hayan pasado sin problemas.

Sin embargo, ambos criterios son independientes de la calidad del producto.

Como se ve, la mayor parte de las veces no existen criterios objetivos para determinar cuando finalizar el esfuerzo de pruebas.

A continuación se dan algunas pautas que pueden utilizarse para definir el término de las pruebas.

Criterio de Término de las Pruebas

Criterio número 1

Metodología específica de pruebas

La idea es contar con metodologías especificas para la generación de casos de
pruebas, tales como, clases de equivalencia, análisis de valores límites, etc. La prueba terminará cuando todos los casos construidos con esta metodología ha-
yan pasado sin problemas.

Problemas de este criterio

1. No son aplicables si no existen métodos definidos
2. No existe certeza de la rigurosidad y correctitud de la utilización de la técnica
3. Está orientado por el método a utilizar y no por los resultados a lograr

Criterio número 2

Por cantidad de errores estimados

La idea es estimar de antemano una cantidad de errores existentes en el programa.
Dado que el número de errores estimado es bastante alto, se puede establecer un porcentaje de errores a encontrar, antes de lo cual no finalizará la prueba.

La estimación puede hacerse en base a datos anteriores de la organización o utilizar datos de la industria.

Si se estima en 10/1000 [errores/loc] después de la inspección de código.

Supongamos una aplicación de 10.000 líneas de código. Esta aplicación potencialmente tendría 100 errores. Aplicando los porcentajes de errores de cada fase vistos en las estadísticas al comienzo del curso, se llega a:

Errores de Requerimientos: 56
Errores de Diseño : 27
Errores de Programación : 7

Criterio de Término de las Pruebas

Criterio número 2

Por cantidad de errores estimados

Si se ha establecido encontrar un 75 % de los errores, entonces la prueba termi-
nará cuando se hayan encontrado:

Errores de Requerimientos	:	42
Errores de Diseño	:	20
Errores de Programación	:	5
Otros	:	8
TOTAL	:	75

Problemas de este criterio

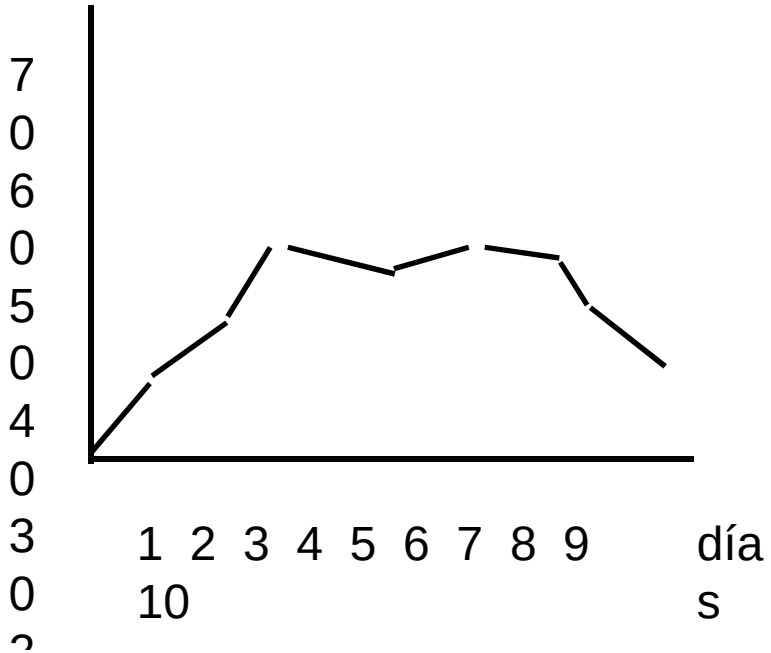
1. Se requiere datos previos para estimar
2. El programa podría contener menos errores que lo estimado. Un poco de sen-
   tido común soluciona este “problema”.
3. El tiempo puede ser más largo que el presupuestado.

Criterio de Término de las Pruebas

Criterio número 3

Por cantidad de errores encontrados por unidad de tiempo

Este método para determinar que la prueba debe terminar consiste en graficar la cantidad de errores encontrados por unidad de tiempo. La prueba termina cuando la cantidad de errores encontrados por unidad de tiempo decrece significativamente.



Criterio de Término de las Pruebas

Criterio número 3

Problemas de este método

1. Las pruebas pueden decrecer por razones que no sean la disminución de errores,
   tales como, baja en la eficiencia de las pruebas.

2. Los casos de pruebas son insuficientes.

3. Exige llevar un registro y gráfico de las pruebas.

> Una combinación de los tres métodos anteriores es una alternativa más eficaz que utilizar sólo uno de ellos.

Métricas del Testing

Métricas de Testing

Un proyecto de Testing debe considerar la recopilación de métricas que permitan conocer el estado y los resultados que se están obteniendo a lo largo del ciclo de vida del proyecto de Testing

Tipificación de defectos

Los defectos que se encuentren debe ser catalogados y registrados. La clasificación depende
del tipo de producto inspeccionado.

Algunos criterios generales de clasificación pueden ser los siguientes:

Requerimientos	Diseño	Software
Omisión	Datos	programa
inconsistencia	Arquitectura	procedimiento
ambigüedad	Rendimiento	software
Infactibilidad	Vistas	básico
Incompletitud		

Métricas de Testing

Para el proceso de Testing de software se recomienda tomar al menos las siguientes métricas:

Cobertura

Cantidad de requerimientos
Número de casos de prueba por requerimiento

Control de Actividades

Cantidad de casos de prueba ejecutados
Número de requerimientos probados

Detección de Defectos

Cantidad de defectos detectados
Cantidad de defectos por tipo (Leve, Grave)

Control de Corrección

Cantidad de Defectos abiertos
Cantidad de defectos corregidos

Plan de Pruebas
y
Plan de Calidad

Plan de Pruebas

El Plan de pruebas es un documento que establece las prácticas específicas de pruebas, recursos y secuencia de actividades relativas un producto, servicio, contrato o proyecto, en particular.

Contenido

1. Objetivos del Plan de Pruebas
2. Breve descripción del producto
3. Criterios de término de las pruebas
4. Responsabilidades de escritura de casos, ejecución y registro de resultados de las pruebas
5. Administración de bibliotecas de casos de pruebas
6. Herramientas de apoyo a las pruebas
7. Especificación de los tipos de pruebas a realizar (Unidad, funcional,..)
8. Planificación de actividades de pruebas
9. Recursos computacionales críticos y riesgos
10. Informes

Plan de Pruebas

1. **Objetivos del Plan de Pruebas**

   Establece para que fines es desarrollado el Plan de Pruebas, sus alcances, entorno de sistemas e información disponible que complementa este plan. Dependiendo de la envergadura del proyecto de testing, puede ser necesario definir cada uno de los puntos siguientes para cada módulo:

2. **Breve descripción del producto**

   Describe en forma resumida las macro funciones del producto, así como su arquitectura de Hw y Sw.

3. **Criterios de término de las pruebas**

   Se identifican los distintos acuerdos que se tendrán presentes para dar por finalizada una prueba.

4. **Responsabilidades de escritura de casos, ejecución y registro de resultados de las pruebas**

   Se identifican las personas que serán las encargadas de cada una de estas actividades. Dependiendo del caso, esta responsabilidad puede ser asumida por una o más personas.

5. **Administración de bibliotecas de pruebas (SCM)**

   Establece la metodología a utilizar para realizar la administración de las piezas de software a testear, asi como de los casos y datos de pruebas.

Plan de Pruebas

6. **Herramientas de apoyo a las pruebas**

   Identificar si se usarán herramientas de software para apoyar algunos tipos de pruebas (Ej. Inspección de código, generación de casos y/o datos, rendimientos, stress)

7. **Especificación de los tipos de pruebas a realizar (Unidad, funcional,..)**

   En esta sección deben identificarse los distintos tipos de pruebas a realizar. Debe contener explícitamente las funciones que serán probadas, los aspectos de diseño considerados, las pruebas de integración y cualquier otra prueba a realizar.
   El día que Ud. sea responsabilizado por un error liberado en el sistema, estará feliz de tener un documento escrito (y firmado) que muestre que esa parte del sistema estaba explícitamente excluida de la prueba.

8. **Planificación de actividades de pruebas**

   Incluir una lista de actividades, fechas y plazos. Dependiendo de la envergadura del proyecto puede construirse una carta Gantt.

9. **Recursos computacionales críticos y riesgos**

   Identificar los recursos, humanos, Hw y Sw que se consideran fundamentales par el éxito del proyecto e identificar el riesgo de no contar con él.

10. **Informes**

    Definir los informes, contenidos y formatos de los reportes a generar en cada actividad.

Plan de Pruebas

Estimaciones de Esfuerzo

Uno de los aspectos más relevantes en la confección de un Plan de Pruebas es el problema de estimar el esfuerzo que demandará realizar las actividades de pruebas.
Es necesario considerar que cada proyecto es diferente, así como cada organización también lo es.
Sin embargo, es conveniente tener en cuenta los siguientes datos:

a. Stephen Brooks estima que el 50% de los costos de un proyecto se destinan a corregir errores
b. Compuware estima que el 50% de los recursos del un proyecto se deben destinar a Testing
c. Caper Jones estima que hay 18 tipos de pruebas, las cuales insumen el siguiente Esfuerzo del total del proyecto:

1a prueba	10 %
2a prueba	15 %
3a prueba	20 %
4a prueba	25 %
5a prueba	30 %
6 a prueba	33 %

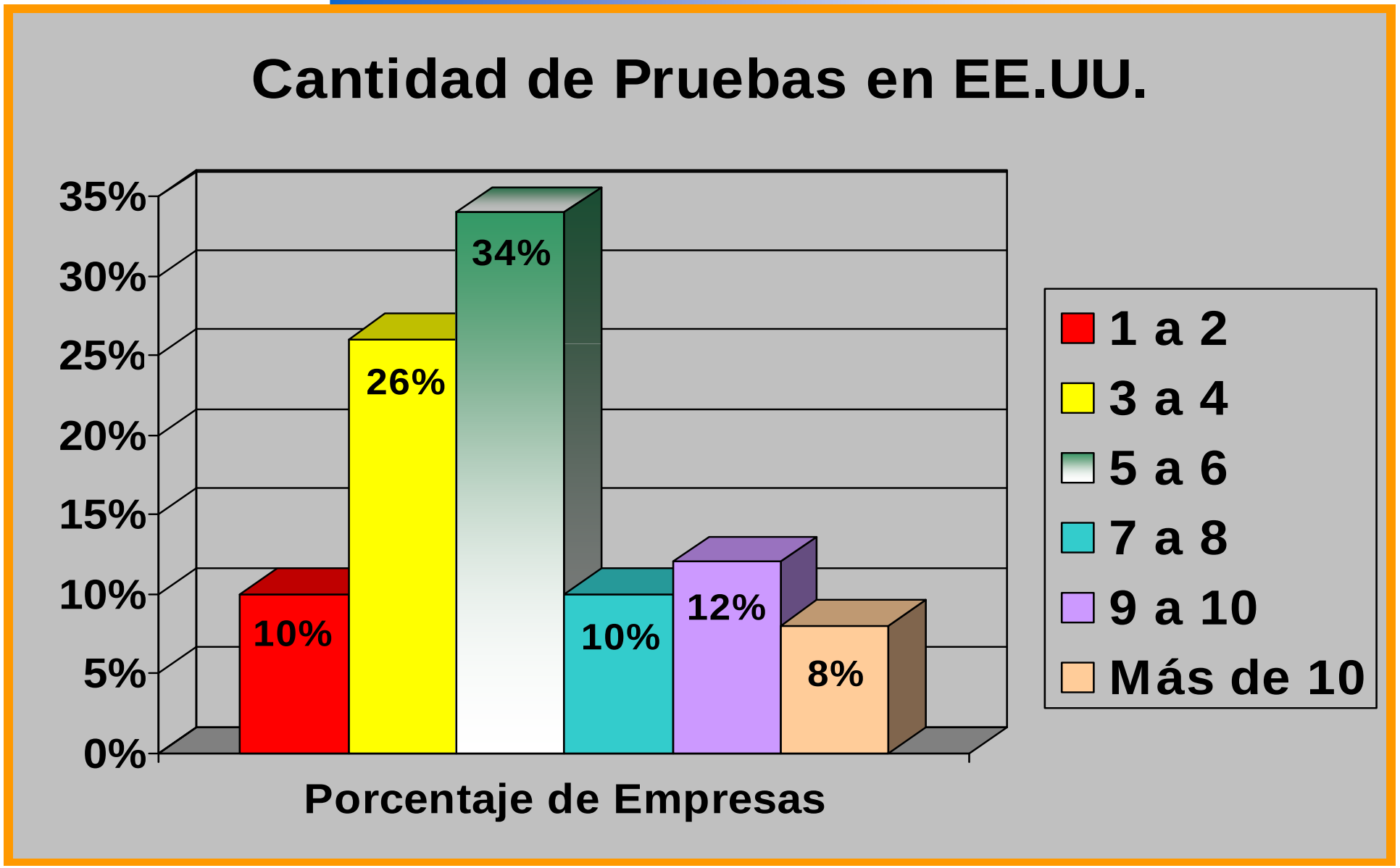
A contar de la 6a prueba en adelante se debe sumar un 3% por cada prueba adicional.

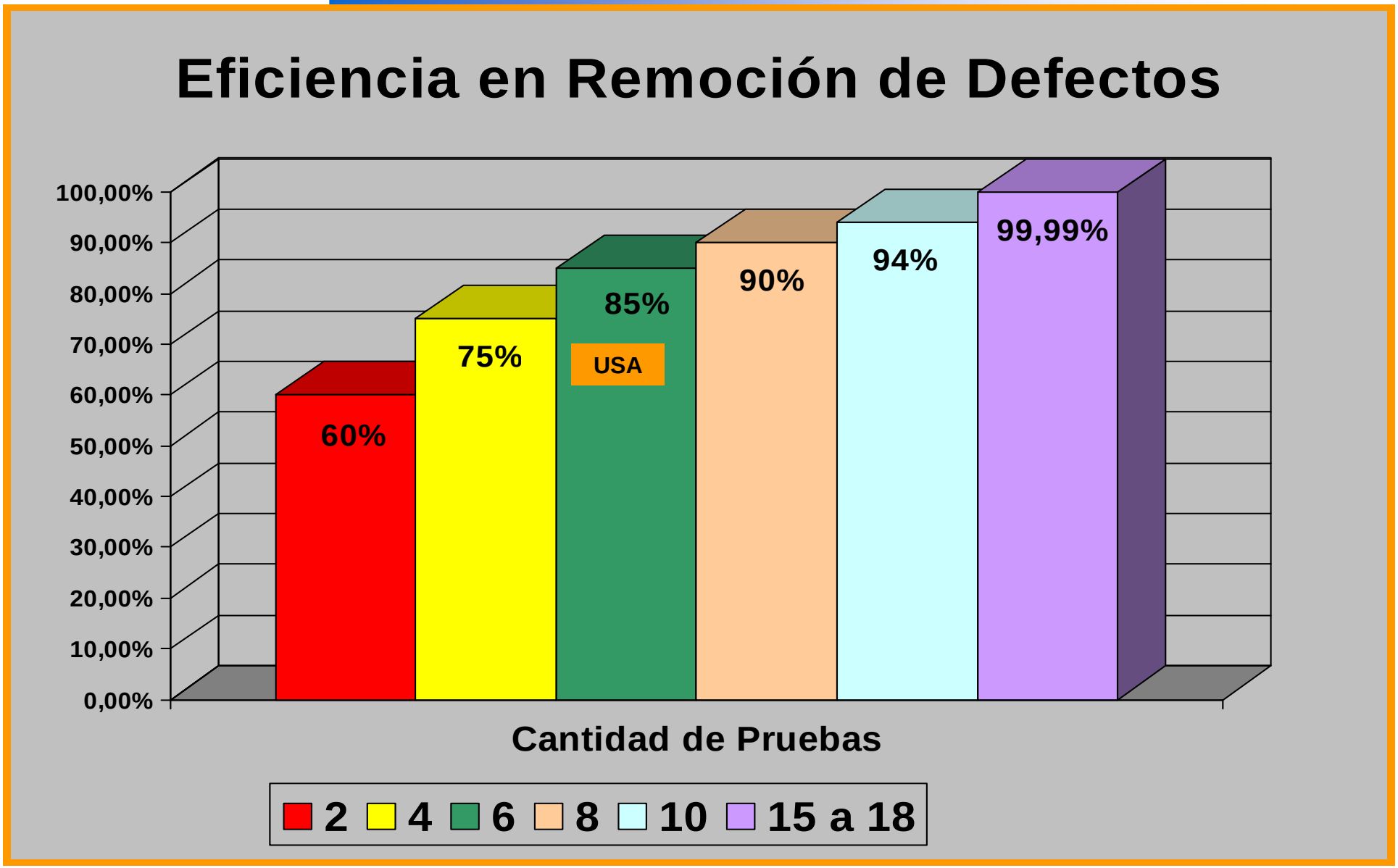
Estimaciones de Eficiencia

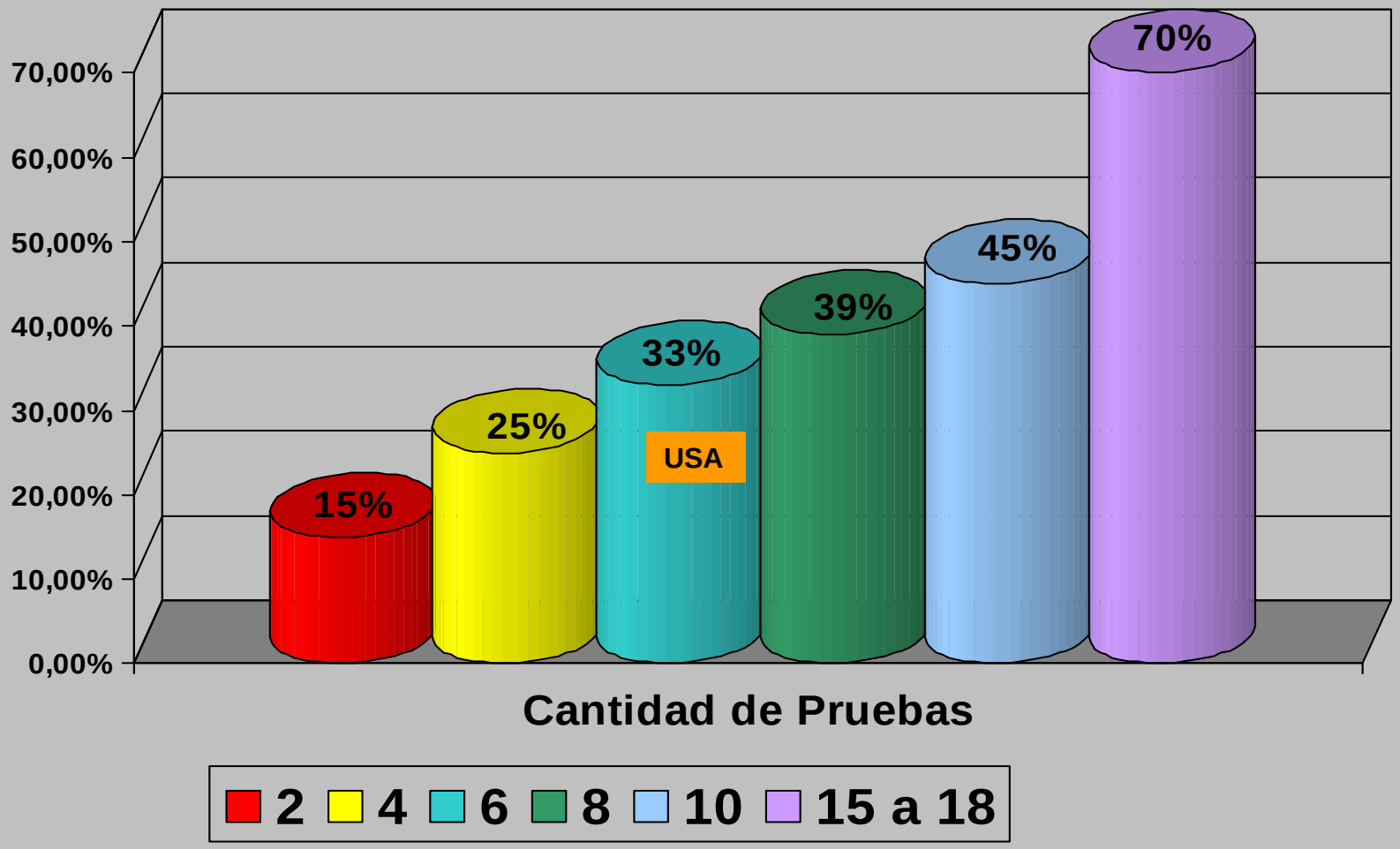
La capacidad de estimar la remoción de defectos, como ya se ha visto, está influida por múltiples factores. Para efectos de entregar consideraciones de caracter generales, a continuación se complementa la tabla de la página anterior agregando estimaciones en relacióna la eficiencia obtenida mediante la relaización de diferentes tipos de pruebas.

Nº de Pruebas	Esfuerzo	Eficiencia acumulada
1a prueba	10 %	50%
2a prueba	15 %	60%
3a prueba	20 %	70%
4a prueba	25 %	75%
5a prueba	30 %	80%
6a prueba	33 %	85%

Nótese que con 6 tipos de pruebas se alcanza en promedio un 85% de eficiencia en la remoción de defectos con un 33% del costo total del proyecto. Esta estimación es derivada de proyectos >= 1000 PF. Fuente: Caper Jones

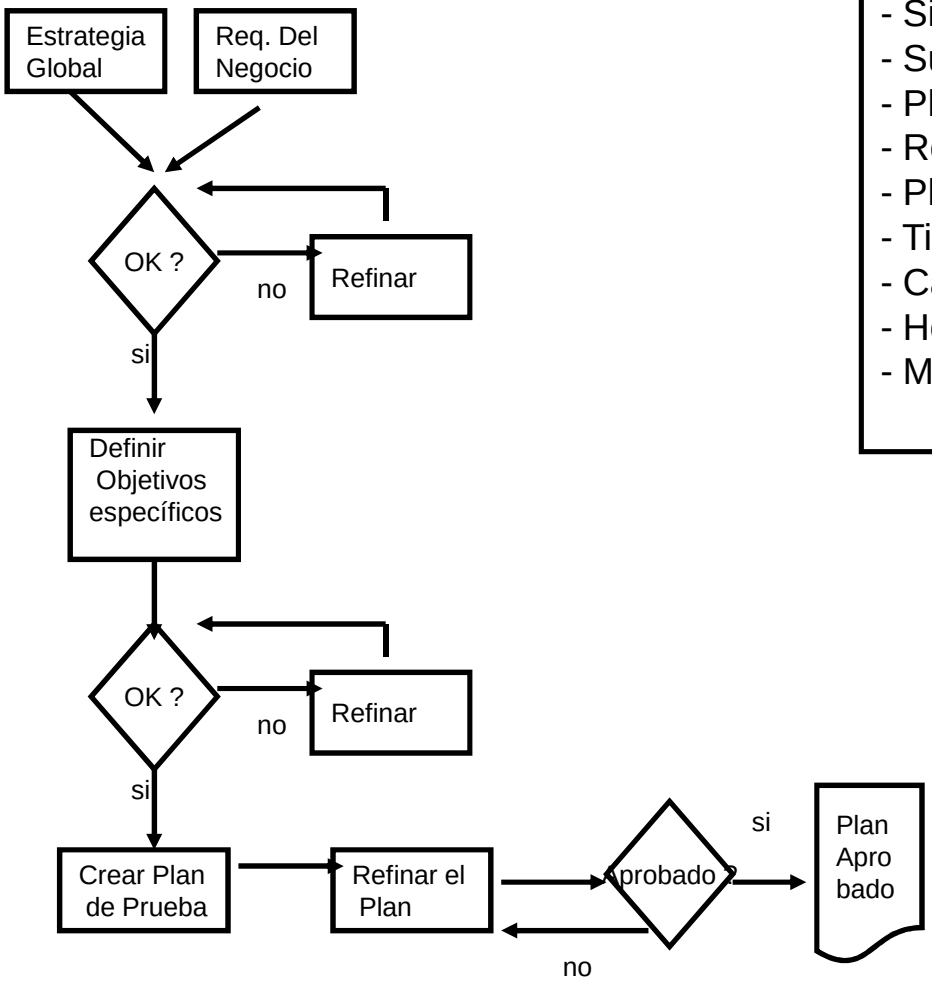

Cantidad de Pruebas en EE.UU.
35%
30%
25%
20%
15%
10%
5%
0%
10%
26%
34%
10%
12%
8%
1 a 2
3 a 4
5 a 6
7 a 8
9 a 10
Más de 10
Porcentaje de Empresas


Eficiencia en Remoción de Defectos
100,00%
90,00%
80,00%
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
60%
75%
85%
USA
90%
94%
99,99%
Cantidad de Pruebas
2
4
6
8
10
15 a 18


Esfuerzo de Testing
70,00%
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
15%
25%
33%
USA
39%
45%
70%
Cantidad de Pruebas
2
4
6
8
10
15 a 18

Plan de Pruebas

Pasos para la confección del Plan de Pruebas



Estrategia Global
- Sistema a probar
- Subsistemas
- Plataformas
- Recursos disponibles
- Plazos deseados
- Tipos de pruebas
- Capacitación
- Herramientas
- Metodologías

Objetivos específicos
- Módulos a probar
- Análisis de transacciones
- Planes por c/subsistema
- Estimaciones
  - Nº Req. estimados
  - Nº Casos Estimados
  - Plazos estimados

Plan de Pruebas
1. Objetivos del Plan de Pruebas
2. Breve descripción del producto
3. Criterios de término de las pruebas
4. Responsabilidades
5. Administración de bibliotecas de casos de pruebas
6. Herramientas de apoyo a las pruebas
7. Especificación de los tipos de pruebas a realizar
8. Planificación de actividades de pruebas
9. Recursos computacionales críticos y riesgos
10. Informes

Plan de Calidad

Plan de Calidad

El Plan de calidad es un documento que establece las prácticas específicas de calidad, recursos y secuencia de actividades relativas a un producto, servicio, contrato o proyecto, en particular. Su confección es similar a lo descrito para el Plan de Pruebas.

Contenido

1. Objetivos del Plan de Calidad
2. Breve descripción del producto
3. Parámetros de calidad cuantitativos
4. Métodos de Control de Calidad a utilizar
5. Herramientas de apoyo a la calidad
6. Planificación de actividades de Control de Calidad
7. Recursos
8. informes

Estrategia de Pruebas

Estrategia de Pruebas

Cada una de la técnicas estudiadas cubren diferentes partes del problema de verificar y validar la calidad de un producto de software. Ninguna de ellas resuelve el problema en su totalidad. Ello hace necesario combinarlas en una estrategia general.

Dada las características del problema, no es posible garantizar un producto sin defectos. Por lo mismo, es necesario adquirir un compromiso razonable de aseguramiento de calidad del proceso, que permita mejorar sustancialmente las posibilidades de obtener un producto que cumpla con una cierta calidad esperada.

Los siguientes criterios pueden ayudar a construir una estrategia de pruebas razonable :

Elementos a considerar en una estrategia de pruebas

1. Verificar las especificaciones de Requerimientos y el Diseño mediante Revisiones
2. Verificar los productos de cada fase mediante Inspecciones
3. Si las especificaciones contienen múltiples combinaciones de entrada, usar Causa y Efecto
4. En todos los casos usar Clases de Equivalencia y Valores Límites
5. Analizar la necesidad de usar Cobertura de Instrucciones y Decisiones, si es que ello no ha sido cubierto por los casos anteriores
6. Determinar necesidades de pruebas de Usabilidad y Web
7. Planificar pruebas de sistemas de acuerdo a las características de la aplicación

Metodología de Pruebas

Metodología de Pruebas

Hasta este momento, se han visto una serie de herramientas y técnicas orientadas a realizar un completo control de calidad de diferentes partes del desarrollo de software. Sin embargo, su aplicación en la práctica requiere de la elaboración de una metodología que integre en un todo coherente los diferentes elementos que están involucrados en la actividad de desarrollo y control de calidad de software.

Previamente, es necesario establecer algunos conceptos de carácter general:

Las pruebas de software deben ser una actividad que está presente a lo largo de todo el ciclo de vida. Asimismo, los casos de pruebas deben ir siendo desarrollados en cada una de las etapas del ciclo de vida.

Metodología de Pruebas

Al comienzo del desarrollo de un sistema es necesario desarrollar un Plan de Calidad que especifique los elementos de calidad a alcanzar por el sistema.

Una de las definiciones de calidad de software más conocidas es la de McCall. Esta definición considera una serie de elementos que permiten establecer las características de un determinado sistema.

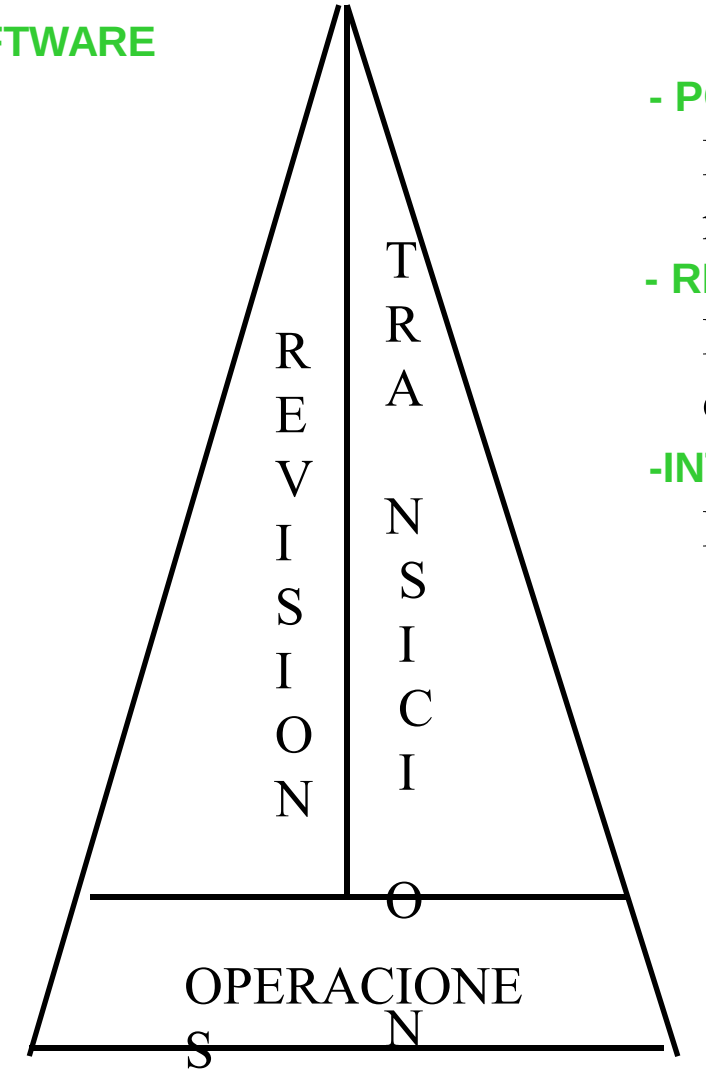
Utilizando la definición referida es posible elaborar un plan de calidad asignando valores cuantitativos a todos o algunos de los factores de calidad definidos por McCall.

Metodología de Pruebas

DEFINICION DE CALIDAD DE SOFTWARE DE McCALL

- **MANTENIBILIDAD**
  Puedo arreglarlo ?
- **FLEXIBILIDAD**
  Puedo cambiarlo ?
- **TESTEABILIDAD**
  Puedo probarlo ?

- **CORRECTITUD :**
  Hace lo que deseo ?
- **CONFIABILIDAD :**
  Es exacto ?
- **EFICIENCIA :**
  Usa el HW correctamente ?
- **INTEGRIDAD :**
  Es seguro ?
- **USABILIDAD :**
  Puedo usarlo ?



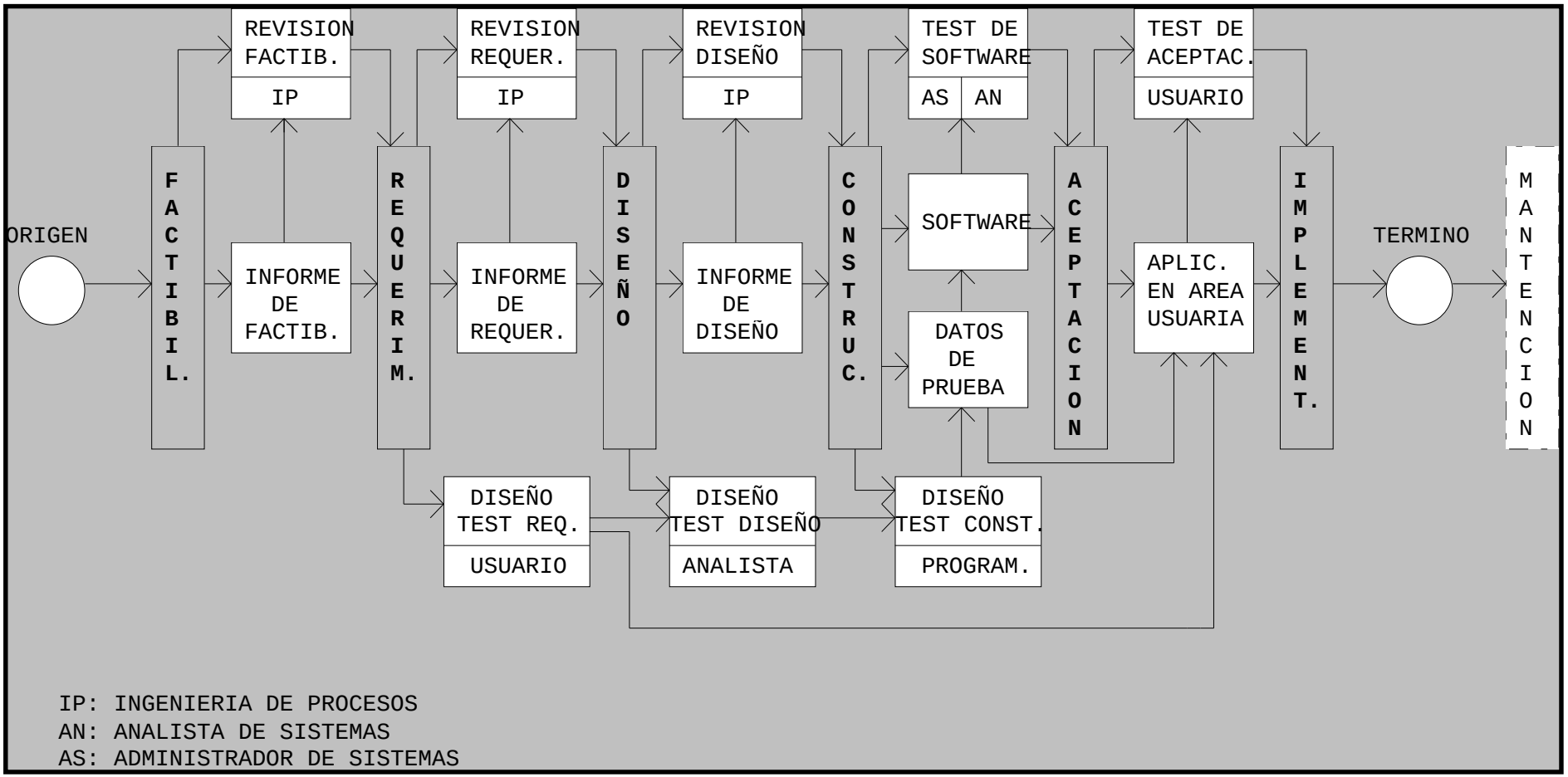
- **PORTABILIDAD**
  Puedo usarlo en otra máquina ?
- **REUSABILIDAD**
  Puedo reusar algo del software ?
- **INTEROPERABIL.**
  Puedo conectarlo

Metodología de Pruebas

Las actividades de Control de Calidad deben formalizarse en un documento en el
que se especifiquen:

- Fases, Actividades y Tareas
- Procedimientos
- Formularios
- Responsables
- Productos
- Estadísticas
- Estándares
- Otros

Metodología de Pruebas



Las pruebas deben hacerse a lo largo de todo el ciclo de vida

Recomendaciones Finales

Recomendaciones Finales

En relación al proceso de pruebas de su organización:

1. Haga una lista de las fortalezas
2. Haga una lista de los problemas
3. Analice las actividades de Verificación
4. Analice las actividades de Validación
5. Describa los riesgos existentes
   - para la organización
   - para las personas
6. Analice los costos que el actual proceso tiene
7. Haga una lista de actividades priorizadas que permitirían mejorar
8. Decida que acciones tomará al respecto

Pruebas de Software

América XXI
Paseo Huérfanos 669, 6°Piso
Santiago - Chile
www. americaxxi.cl
Tel: 56-2-633 5415 – Fax:56-2-632 6940

Anexos

Anexo 1. Factores de Calidad de software

Corrección: Es el grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el cliente.

Fiabilidad: Es el grado en que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida.

Eficiencia: La cantidad de recursos computacionales y de código requeridos por un programa para llevar a cabo sus funciones.

Integridad: El grado en que puede controlarse el acceso al software o a los datos por personal no autorizado.

Facilidad de Uso: El esfuerzo requerido para aprender, trabajar con, preparar la entrada e interpretar la salida de un programa.

Facilidad de Mantenimiento: El esfuerzo requerido para localizar y arreglar un error en un programa.

Flexibilidad: El esfuerzo requerido para modificar un programa operativo.

Anexo 1. Factores de Calidad de software

Facilidad de Prueba: Es el esfuerzo requerido para probar un programa de forma que asegure que realiza su función requerida.

Portabilidad: El esfuerzo requerido para transferir el programa desde un hardware y/o entorno de sistemas, a otro.

Reusabilidad: El grado en que un programa (o partes de un programa) se pueden reusar en otras aplicaciones.

Interoperabilidad: El esfuerzo requerido para acoplar el sistema a otro.

Anexo 2. Métricas de Calidad de software

Facilidad de Auditoría: La facilidad con que puede comprobarse la conformidad con los estándares.

Exactitud: La precisión en los cálculos y el control.

Normalización de las comunicaciones: El grado en que se utilizan el ancho de banda, los protocolos y las interfaces estándar.

Completitud: El grado en que se ha conseguido la total implementación de las funciones requeridas.

Concisión: Lo compacto que es el programa en términos de lineas de código.

Consistencia: El uso de un diseño uniforme y de técnicas de documentación a lo largo del proyecto de desarrollo de software.

Estandarización de los datos: El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.

Anexo 2. Métricas de Calidad de software

Tolerancia de errores: El daño que se produce cuando el programa encuentra un error.

Eficiencia en la ejecución: El rendimiento en tiempo de ejecución de un programa.

Facilidad de expansión: El grado en que se puede ampliar el diseño arquitectónico, de datos o procedural.

Generalidad: La amplitud de aplicación potencial de los componentes del programa.

Independencia del hardware: El grado en que el software es independiente del hardware sobre el que opera.

Instrumentación: El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen.

Modularidad: La independencia funcional de los componentes del programa.

Facilidad de operación: La facilidad de operación de un programa.

Anexo 2. Métricas de Calidad de software

Seguridad: La disponibilidad de mecanismos que controlen o protejan los programas o los datos.

Autodocumentación: El grado en que el programa fuente aporta documentación significativa.

Simplicidad: El grado en que un programa puede ser entendido sin dificultad.

Independencia del sistema de software: El grado que el programa es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones ambientales.

Facilidad de traza: La posibilidad de seguir la pista de la representación del diseño o de los componentes reales del programa hacia atrás, hacia los requerimientos.

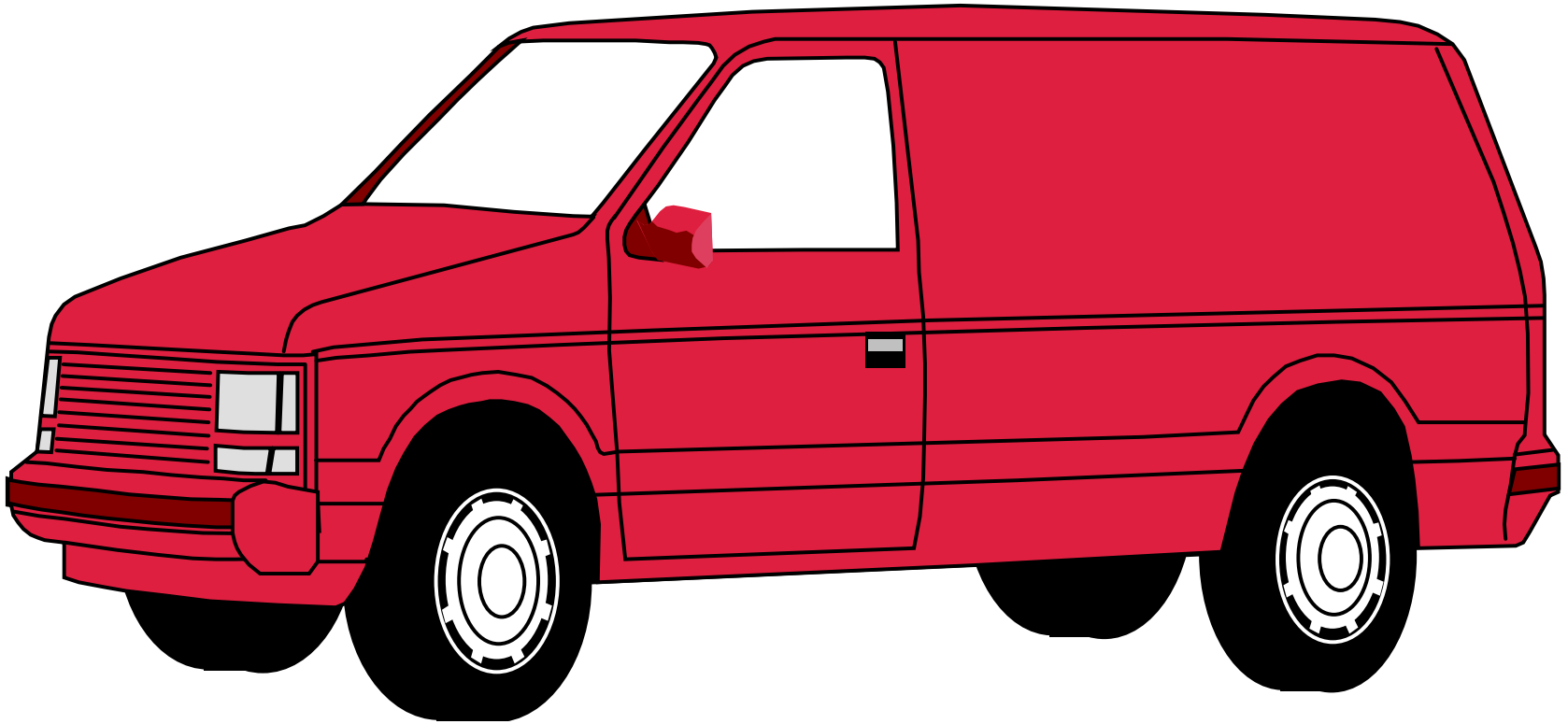
Formación: El grado en que el software ayuda para permitir que nuevos usuarios utilicen el sistema.

Solución al problema del triángulo (Ejercicio Nº1)

1. Un triángulo escaleno válido ?
2. Un triángulo equilátero válido ?
3. Un triángulo isósceles válido ?
4. Tres casos correspondientes a las tres permutaciones de un triángulo escaleno, por ejemplo: 3,3,4; 343; y 4,3,3.
5. Un caso donde por lo menos un lado tenga valor nulo ?
6. Un caso en el cual un lado tenga valor negativo ?
7. Un caso con tres enteros mayores que cero tales que la suma de dos de ellos sea igual tercero ?. Si el programa responde que éste es un triángulo escaleno será un error.
8. Tres casos de prueba correspondientes a las permutaciones del caso anterior ?
9. Un caso de prueba con tres enteros mayores que cero tales que la suma de las longitudes de dos lados sea menor que la del tercero?
10. Tres casos de prueba correspondientes a las permutaciones del caso anterior?
11. Un caso de prueba con todos sus lados iguales a cero?
12. Un caso con valores decimales?
13. Un caso con menos de tres lados?
14. Especificó para cada caso la salida esperada?

Solución al problema del vehículo (Ejercicio Nº2)

¿Incluyó los problemas de _____ y de _____ en sus casos de pruebas ?



Solución a la Inspección de Requerimientos (Ejercicio Nº3)

1. La cantidad de dólares ingresada deberá ser sumada al Total_ Año a la fecha. Este valor debe ser positivo.
   - ¿Ingresada desde dónde? (terminal, diskette, otro)
   - ¿Sólo los dólares?, ¿Que pasa con los centavos?
   - ¿Cuál valor debe ser positivo? ¿El total o la cantidad ingresada?
   - ¿Cuál monto? ¿Monto es sinónimo de cantidad?

2. Hay tres condiciones de error que producen un mensaje. El dato es no numérico, tiene un valor negativo o excede el valor máximo.
   - ¿Qué significa “producir”?
   - ¿Un mensaje o más de uno?
   - ¿Cuál es el mensaje?
   - ¿A que dato se refiere?
   - ¿Las condiciones enunciadas son las causas de error o los valores correctos?
   - ¿Cuál es el valor máximo?

3. Si el monto es superior a US$100.000 se debe rechazar la transacción.
   - ¿Que se entiende por transacción? ¿El hecho de ingresar la cantidad? ¿Se acepta la cantidad, pero, se realiza una acción distinta?
   - ¿Rechazar implica enviar algún mensaje?
   - Falta el Else

Solución a la Inspección de Requerimientos (Ejercicio Nº3)

4. Al final del proceso se debe actualizar la base de datos con el total anual calculado.
   - ¿Cuándo es el final del proceso?
   - ¿Se refiere al proceso de cada transacción o de todas las transacciones?
   - ¿Cual base de datos?
   - ¿Que campo o campos de la base de datos?
   - ¿Cuál es el total anual acumulado?
   - ¿Qué significa actualizar? ¿Reemplazar, sumar, restar?

5. Si todo estuvo correcto, imprimir totales de control y terminar.
   - ¿Que significa “Si todo estuvo correcto”?
   - ¿Cuáles totales de control?
   - ¿Dónde se imprimen?
   - ¿Qué significa terminar? ¿pasar el control a otro programa? ¿Informar al operador?
   - Falta el Else.

6. Nota: Las operaciones en UF se tratan de manera similar a como se especificó para los dólares.
   - ¿Operaciones es sinónimo de transacciones?
   - De manera similar es ambiguo.

Bibliografía

1. ISO 9000-3 Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software. ISO Geneve - Suitzerland.
2. Normas Chilenas de la Serie ISO 9000. (INN-Chile).
3. IEEE Standard for Software Unit Testing. ANSI/IEEE Std 1008-1987
4. IEEE Standard for Software Verification and Validation Plans. ANSI/IEEE Std 1012-1986
5. Improving Software Quality. Lowell Jay Arthur - John Wiley & Sons, Inc., NY. 1993.
6. Ingeniería del Software. Roger S. Pressman - McGraw-Hill. 1990.
7. Managing The Software Process. Watts S. Humphrey. Addison Wesley Co.
8. The Art of Software Testing. Glenford J. Myers, Ed.John Wiley & Sons, Inc., NY. (Versión en español: Ed. Ateneo. Buenos Aires).
9. Software Audits. Unix Review, USA. Octubre 93
10. Breaking through the Testing Barrier. The Journal of Quality Assurance Institute. Octubre 1993.
11. The Quest for Quality: Computer-Aided Software Testing. Case Strategies, Ma USA, 1992.
12. Software Testing (American Programmer Abril 1994). Ed. Cutter Information Co.
13. Defect Detectives. The Journal of the Quality Assurance Institute. Octubre 1994.
14. Debugging thr Development Process. S. Maguire. Ed Microsoft Press. 1994
15. The Craft of Software Testing. B. Marrick. Ed. Prentice Hall. 1995
16. Software Quality. Caper Jones. Ed. Int. Thompson Computer Press. 1997
17. Applying Software Metrics. Paul Oman. Ed. Ieee Computer Society Press. 1997
18. Sofyware Fault Injection. Voas y Macgraw. Ed Wiley & Sons 1998
19. The Web testing Handbook.Stevn Splaine. STQE Publishing. 2001
20. Software Guru. Año 02/06.software Testing