



## Diseño de Sistemas basado en TICS

### Construcción

Codificación

Pruebas

Control de calidad

Entrega y aceptación

Aldo Di Biase Friedmann



## Diseño de Sistemas basado en TICS

### Codificación

Codificación

Consideraciones

Perosnal

Aldo Di Biase Friedmann

## Codificación

- Corresponde a la programación propiamente tal
- Es la etapa que incluye más personas y habitualmente es la más larga

## Consideraciones

- Uso eficiente de la herramientas
  - Normalmente es una decisión de diseño, pero se completa y detalla como parte de la construcción
- Uso de estándares y nomenclatura
- El usuario no participa, por lo que se encuentra “alejado”
  - Es importante mantenerlo actualizado del avance

## Entregas parciales

- Aunque el sistema se considere en forma completa como una sola versión (recordar el modelo espiral de desarrollo) se pueden hacer entregas parciales
- Por ejemplo construir y entregar primero los programas que permitirán ingresar la información básica del sistema y luego los relacionados con los procesos propiamente tales

## Entregas parciales

- Ejemplo U-Cursos
  - Hacer primer los programas que permiten ingresar profesores y alumnos
  - Luego los que permiten crear secciones para un semestre específico
  - En tercer lugar los programas que permiten ingresar las evaluaciones, subir material y todo lo relacionado con el proceso docente
  - Finalmente se construyen los que permiten obtener estadísticas y otra información agregada

## Personal

- Los programadores normalmente no son las mismas personas que han hecho el diseño
  - A veces el diseñador realiza alguna parte crítica del código
- Por cada diseñador se pueden tener 2 a 3 programadores
- Pueden haber programadores diferentes, de acuerdo a las herramientas (lenguajes) que se utilicen

## Diseño de Sistemas basado en TICS

### Pruebas

Importancia

Personal

Tipos de pruebas

Metodología de pruebas

## Importancia de las pruebas

- En las pruebas se verifica que el sistema cumpla con lo que se espera de él
- Hasta la codificación todo el desarrollo ha sido en papel, recién en ese momento se “tiene” el sistema
- Es común que en la codificación se comentan errores
- También es común que haya que hacer ajustes al diseño

## Importancia de las pruebas

- Por lo anterior es crítico hacer una buena etapa de pruebas
- Además es una etapa poco valorada hasta hace pocos años

## Consideraciones importantes

- Las pruebas son para encontrar errores, no para encontrar todo bueno
- Las pruebas las hace una persona diferente a quien haya programado
- Es normal que en las pruebas se encuentren formas “mejores” de hacer las cosas
  - Esta situación debe ser planeada desde el principio

## Personal

- Existen dos posibilidades
  - El propio equipo de desarrollo
  - Un grupo especializado
- Para desarrollos pequeños (o en empresas pequeñas) el propio grupo de desarrollo realiza las prueba
  - Ventaja: Conoce el sistema
  - Ventaja: Puede colaborar con la solución
  - Desventaja: Puede hacer una prueba sesgada
- En organizaciones mayores existe un grupo de pruebas
  - Ventaja: Especialización
  - Ventaja: Imparcialidad
  - Desventaja: Debe conocer el sistema

## Personal

- En algunos casos se pueden utilizar ambos grupos
  - Las pruebas unitarias y de integración las realiza el grupo de diseño
  - Las pruebas de sistema las realiza un grupo especializado
- De hecho el grupo de desarrollo (y/o el de construcción) debe probar por si mismos las aplicaciones antes de entregarlas para su prueba por grupos externos

## Tipos de pruebas

- Unitarias
  - Se revisa cada componente por separado
- De integración
  - Se revisa la integración con otros sistemas
- De sistema
  - Se revisa el funcionamiento del sistema como un todo
- De performance / carga
  - Se revisa que la performance del sistema sea la esperada
- De aceptación
  - Es la revisión del usuario

## Tipos de pruebas

- Caja negra
  - No se ve el código de la aplicación
- Caja blanca
  - Se ve y revisa el código

## Diseño de Sistemas basado en TICS

### Metodología de pruebas

Revisión de código  
Prueba de escritorio  
Clases equivalentes  
Escenarios de uso



## Revisión de código

- Consiste en que una (o más) persona revisa el código y hace sus comentarios
- Es la forma más poderosa de capturar errores
  - No sólo es capaz de detectar errores en la aplicación presente, si no puede “preveer” problemas de mantención
- Existe la versión informal en que se hace “revisión peer to peer”
  - Un programador revisa el código de otro

## Revisión de código

- La versión más potente es la formal
  - Tiene una estructura y roles bien definidos
    - Autor
    - Moderador
    - Lector
    - 2 a 4 participantes
  - El lector lee el código
  - Todos entregan sus comentarios
  - No es un tema personal, si no del código realizado
  - Surgen, normalmente, temas anexos que se dejan anotados para ser abordados más adelante

## Ejemplo: Ecuación de segundo grado

```
Sub Ecuacion()  
Dim a, b, c As Double  
Dim y As Double  
Dim x1, x2 As Double  
  
y = b * b - 4 * a * c  
If y >= 0 Then  
    x1 = (-b + sqrt(y)) / 2 * a  
    x2 = (-b - sqrt(y)) / 2 * a  
Else  
    x1 = (-b + sqrt(-y)) / 2 * a  
    x2 = (-b - sqrt(-y)) / 2 * a  
End If  
  
End Sub
```

## Prueba de escritorio

- Consiste en ejecutar el código “a mano”
- Era muy usada cuando la programación y compilación era cara
- Hoy en día se usa una versión modificada
  - Se ejecuta el código paso a paso
  - Se revisa el valor de las variables, acciones en pantalla, etc.
  - Permite detectar inconsistencias y otros errores
- Se usa en conexión con otras como las clases equivalentes

## Clases equivalentes

- Se utiliza tanto en pruebas de caja negra o caja blanca
- Consiste en identificar los diferentes conjuntos de valores que pueden tomar los datos para generar resultados equivalentes
- Ejemplo: porcentajes
  - Clase 1: Menores a cero
  - Clase 2: cero
  - Clase 3: mayor a cero y menor a 100
  - Clase 4: 100
  - Clase 5: mayores a 100

## Clases equivalentes

- Al considerar todas las variables aparece un conjunto de todas las clases que deben ser probadas
- Es muy importante considerar los casos de borde
  - Generalmente allí hay problemas no probados
- Las clases no se limitan sólo a variables
  - Por ejemplo una clase podría ser cuando no es posible conectarse a la base de datos
- Deben considerarse todas las condiciones en que podría operar el sistema y las “cosas raras” que podría hacer el usuario

## Clases equivalentes

- Para cada prueba se tiene un resultado esperado
- Puede ser un mensaje de error si la clase no es permitida
- Puede ser el valor final después de los calculos

## Escenarios de uso

- Las pruebas anteriores están orientadas a las pruebas unitarias y de integración
- Para las pruebas de sistema y de aceptación se usan los escenarios de uso
- En general son pruebas de caja negra

## Escenarios de uso

- Un escenario de uso es una secuencia completa de eventos o sucesos
- Ejemplo:
  - El cliente compra algo y se entrega en su oficina
  - El cliente compra algo y lo viene a buscar
  - El cliente compra algo y nunca lo retira
  - El cliente compra algo y luego no lo quiere

## Escenarios de uso

- Al igual que en el caso de las clases equivalentes, considera todas las combinaciones posibles
- También tiene considerados los resultados esperados para cada prueba
  - Y para cada paso de la prueba

## Herramientas de prueba

- Existen herramientas que permiten automatizar las pruebas
  - Se repiten los mismos casos cada vez que sea necesario
  - Pueden capturar la operación normal para generar las clases equivalentes y/o los escenarios de uso
    - Debe actuarse con cuidado en este caso:
      - Muchas clases repetidas
      - Clases que no se ejecutan
  - Muy útiles para pruebas de performance y de carga
    - Automatizan la captura de datos
    - Permiten simular una gran cantidad de usuarios

## Diseño de Sistemas basado en TICS

### Administración de las pruebas

Plan de pruebas

Proceso de prueba

Documentación de errores y su solución

# Plan de pruebas

- Es el documento que guía las pruebas y las organiza
- Es confeccionado por el diseñador o por un equipop especial
- Contenido:
  - Objetivo de las pruebas
  - Tipos de pruebas
  - Condiciones de prueba
  - Datos de prueba
  - Resultados esperados
  - Forma de documentar y resolver los errores

# Ejemplo

Paso	Dato a ingresar	Resultado esperado	Resultado real	Observaciones
1	Rut= 1-9	Mensaje “No está registrado		
2	Rut= 8505004-4	Desplegar nombre		
3	Rut= 8505004-3	Mensaje “Digito verificado inválido		

## Proceso de pruebas

- Es importante acordar a priori el proceso que se seguirá para las pruebas
  - Forma en que se definen las pruebas
  - Forma en que se comunican los resultados
  - Forma en que se comunican las correcciones
  - Forma en que se documentan los errores
  - Forma en que se revisarán en forma posterior
    - Para resolver la causa de los problemas

## Documentación de errores

- Normalmente se usa una planilla complementaria al plan de pruebas
- Se van colocando todas las acciones que se han hecho para resolverlo
  - Por ejemplo, pruebas en ambiente de desarrollo, cambios realizados, etc
- Una vez revisado se realiza una nueva prueba (de regresión) para comprobar que efectivamente se haya resuelto



## Documentación de errores

- Además se debe realizar un control de las versiones del código
  - Existe una versión antes del cambio (que genera el error)
  - Existe una versión después del cambio (que se pasa a prueba)

## Revisión de errores

- Los documentos de plan de prueba y documentación de errores servirán de base para una revisión detallada de los mismos
- Esta revisión está orientada a encontrar la fuente de los errores y poder resolverla
  - Ejemplo: Se puede encontrar que un programador tiene “malas prácticas” lo que podría implicar que requiere capacitación
  - Ejemplo: Mal uso de alguna herramienta, por ejemplo al definir un objeto no identifica las variables de cada método

### Caso especial: Se implementa un producto

Aldo Di Biase Friedmann

## Implementación de producto

- En este caso no se tiene una construcción desde cero
- Dependiendo de la situación se tendrá que es sólo la modificación de parámetros, pero en muchos se requiere codificar adaptaciones
- La parametrización y la codificación debe ser probada
  - Se usan las mismas herramientas y técnicas anteriores
  - Además, puede ser útil la resolución de dudas ofrecida por el fabricante

Aldo Di Biase Friedmann  
Página N° 36

Diseño de Sistemas basado en TICS  
Universidad de Chile

# Conceptos de control de calidad

Aldo Di Biase Friedmann

## Control de calidad

- Control de calidad de productos masivos
- Se basa en dos pilares
  - Cumplimiento de estándares
  - Superar las expectativas de los clientes

Aldo Di Biase Friedmann  
Página N° 38

Diseño de Sistemas basado en TICS  
Universidad de Chile

## Control de calidad de proyectos

- Un proyecto se realiza una sólo vez y no es un producto masivo
- ¿Cómo medir la calidad? ¿Cómo asegurarse que está bien hecho?
- Preocupaciones
  - Plazo
  - Costo
  - Cumplimiento de requisitos

## Control de calidad de proyectos

- Luego en un proyecto, la calidad se debe ir verificando de la mano con la ejecución del mismo

## Estándares de calidad para proyectos TIC

- Norma ISO 9000
  - Es una norma general de calidad, no sólo para proyectos
  - Pero existe una versión específica para SW
  - Es una norma ISO, reconocida mundialmente
  - Se cumple o no se cumple
  - Un organismo externo entrega la certificación

## Estándares de calidad para proyectos TIC

- Modelo de madurez CMM
  - Actualmente está en su versión 2
  - Ideada específicamente para el desarrollo de SW
  - Modelo de madurez en base a una serie de variables
    - Es una estrategia más usada en la actualidad que las “todo o nada”
  - Basada en autoevaluación
  - Los niveles superiores son más exigentes que ISO 9000

## Bases de los estándares de calidad

- Planificación
- Documentación
- Análisis del proceso
  - Errores y aciertos

## Diseño de Sistemas basado en TICS

Construcción

Codificación

Pruebas

Control de calidad

Entrega y aceptación