**Capitulo 29 - Sala Limpia**

Es un enfoque que resalta la necesidad de construir la corrección en el software. En lugar del clásico ciclo de análisis, diseño, código, prueba y depuración, este enfoque sugiere un punto de vista diferente.

En qué consiste:

* Evitar la dependencia de costosos procesos de eliminación de defectos al escribir los incrementos de código correctos la primera vez y verificar su corrección antes de ponerlo a prueba.

**Enfoque de Sala Limpia**

* El proceso de sala limpia destaca el rigor en la especificación y el diseño.
* La verificación formal de cada elemento de diseño mediantes pruebas de corrección con bases matemáticas.
* Es un modelo de proceso que elimina los defectos antes de que puedan generar peligros serios.
* Los errores implican la reelaboración. Ya que estos llevan tiempo y aumentan los costos.
* Es eficaz en cuanto a costo y tiempo.
* Evita la introducción de defectos de producción.
* En vez de fabricar un producto y luego eliminar los defectos, es todo lo contrario.
* Demanda la disciplina requerida para eliminar los errores en la especificación y el diseño y luego fabricarlo en forma “LIMPIA”

Filosofía surgida en 1980 la cual no obtuvo gran difusión. Razones:

* Metodología demasiado teórica, matemática y radical.
* No está a favor de una prueba unitaria de parte de los desarrolladores.
* La madurez de la industria de desarrollo del software.

Los beneficios superan con mucho la inversión requerida para superar la resistencia cultural ubicada en el centro de estas preocupaciones.

**Estrategia de Sala Limpia**

Este enfoque utiliza una versión especializada del modelo de proceso incremental.

Mediante pequeños equipos de trabajo se desarrolla una “línea de incrementos de software”. Conforme cada incremento se certifica se integra en el todo.Por ende, la funcionalidad del sistema crece con el tiempo.

Tareas de sala limpia para cada incremento:

**Planificación del Incremento:** Se desarrolla un plan de proyecto, se crea la funcionalidad de cada incremento, su tamaño proyectado y un plan de desarrollo de sala limpia.

**Recopilación de requisitos: Se** elabora una descripción más detallada de los requisitos del cliente.

Especificación de la estructura de cajas: Se utiliza estructuras de caja para describir la especificación funcional. Estas cajas aíslan y separan la definición de comportamiento, datos y procedimientos en cada grado de refinamiento.

**Diseño formal:** ya que se utiliza la estructura de cajas, el diseño s una extensión natural y uniforme de la especificación.

**Verificación de la corrección**: Se lleva a cabo una serie de rigurosas actividades de verificación de la corrección en el diseño y luego en el código.

**Generación de código, inspección y verificación**: Las especificaciones de estructura de cajas se traducen al lenguaje de programación apropiado. Luego se verifica y corrige el código fuente.

**Prueba estadística de la utilización:** Se ejecutan una serie de pruebas derivadas de una muestra estadística de todas las posibles ejecuciones de programa por parte de todos los usuarios.

**Certificación:** Una vez verificado, el incremento se certifica y queda listo para su integración.

**¿Qué hace diferente a la Sala Limpia?**

Representa el primer intento práctico de someter el proceso de desarrollo de software al control estadístico de la calidad con una estrategia bien definida para la mejora continua de los procesos.

Diferencias con métodos convencionales:

* Emplea en forma explícita el control estadístico de la calidad.
* Verifica las especificaciones del diseño utilizando una prueba de corrección basada matemáticamente.
* Implementa técnicas de prueba con una alta probabilidad de descubrir errores de alto impacto.

Elimina al papel de la prueba unitaria y la depuración y reduce drásticamente la cantidad de pruebas que realiza el desarrollador del software.

**Especificaciones Funcionales**

La ingeniería del software de sala limpia cumple con los principios de análisis operativo empleando un método llamado especificación de estructura de cajas. Una “caja” encapsula al sistema (o algún aspecto de éste) en algún grado de detalle. Por medio de un proceso de elaboración o refinamiento en niveles, las cajas se refinan en una jerarquía donde cada una tiene transparencia referencial.

Se utilizan tres tipos de cajas:

La ***Caja Negra***especifica el comportamiento de un sistema o de una parte de éste. El sistema (o parte de él) responde a estímulos específicos (eventos).

La***Caja de estado***encapsula los datos de estado y servicios (operaciones), se representan las entradas a la caja de estado (estímulos) y las salidas (respuestas).

La***Caja Transparente****,* las funciones de transición que implica la caja de estado se definen en la caja transparente. En otras palabras, contiene el diseño de procedimiento para la caja de estado.

El refinamiento de la estructura de cajas y la verificación de la corrección ocurren simultáneamente.

**Especificación de Caja Negra**

* Una especificación de caja negra describe una abstracción, estímulos y respuesta. La función se aplica a una sucesión de entradas (estímulos) y las transforma en una salida.

**Especificación de Caja de Estado**

* La caja de estado utiliza una abstracción de datos para determinar la transición hacia el siguiente estado y la acción (respuesta) que ocurrirá como consecuente la transición.

**Especificación de Caja Transparente**

* La especificación de caja transparente está cercanamente relacionada con el diseño de procedimientos y la programación estructurada.

**Diseño de Sala Limpia**

El enfoque de diseño utilizado en la ingeniería del Software de Sala Limpia utiliza con amplitud la filosofía de “programación estructurada”.

Las funciones de procesamiento básico refinadas utilizando una expansión progresiva de funciones matemáticas en estructuras de conectivos lógicos y subfunciones.

**Refinamiento y verificación del diseño**

Cada especificación de caja transparente representa el diseño de un procedimiento necesario para lograr una transición de caja de estado.

En cada nivel de refinamiento el equipo realiza una verificación formal de corrección. Se anexa un conjunto de condiciones de corrección genéricas a las estructuras de programación estructurada.

Cada vez que una caja transparente se refina al siguiente detalle se aplican dichas condiciones.

Probar que un diseño es correcto requiere, primero, identificar todas las condiciones y luego probar que cada una toma el valor booleano adecuado. A estas condiciones se les llama subpruebas.

**Ventajas de la verificación del diseño**

***Permiten al equipo de sala limpia verificar cada línea de diseño y código***: Los equipos pueden realizar la verificación por medio del análisis y la discusión en grupo sobre la base del teorema de corrección.

***Resultan en un nivel de defecto cercano a cero:*** Cada miembro deberá estar de acuerdo en que cada condición es correcta. Ya que un error es posible si cada miembro del equipo verifica incorrectamente una condición.

***Es escalable****:* todo software tiene procedimientos de caja transparente de nivel superior compuesto por estructuras de sucesión, iteración e iteración. Las cuales invocan un gran subsistema con miles de líneas de código, que a la vez tienen sus propios procedimientos, etc. Las estructuras de nivel superior e inferior se verifican de la misma forma.

***Produce mejor código que la prueba unitaria:***El enfoque de sala limpia puede verificar cualquier efecto posible sobre todos los datos porque, mientras un programa puede tener muchas trayectorias de ejecución, solo tiene 1 función.

**Pruebas de Sala Limpia**

La estrategia y las tácticas de las pruebas de sala limpia son diferentes de los enfoques de pruebas convencionales.

Los Métodos Convencionales generan un conjunto de casos de prueba para descubrir errores de diseño y codificación.

La meta de las pruebas de Sala Limpia es validar los requisitos de software demostrando que una muestra estadística de casos de uso se ha ejecutado exitosamente

**Pruebas Estadísticas de Uso**

Equivale a probar el software en la forma que los usuarios intentarían usarlo. Esto se logra si los equipos de sala limpia determinan una distribución de probabilidad de uso para el software.

La especificación (caja negra) de cada incremento del software se analiza para definir un conjunto de estímulos (entradas o eventos) que provocan el cambio de comportamiento del software.

Los casos de prueba se generan para cada conjunto de estímulos de acuerdo con la distribución de probabilidad de uso.

Al usar tiempos de intervalo, el equipo de certificación tiene la posibilidad de calcular el tiempo medio entre falla (TMEF). Si se lleva a cabo una larga secuencia de pruebas sin fallas, el TMEF es bajo y es probable que la fiabilidad del software sea alta.

**Certificación**

Las técnicas de verificación y prueba llevan a componentes de software (e incrementos completos) que pueden certificarse.

En Sala Limpia la certificación implica que la fiabilidad (medida en TMEF) se especifica para cada componente.

Los componentes de software reutilizables se pueden almacenar junto con sus escenarios de uso, estímulos de programas y distribuciones de probabilidad. Cada componente tiene una fiabilidad certificada.

El enfoque de la certificación involucra 5 pasos:

1. Creación de escenarios de uso.
2. Especificación de perfiles de uso.
3. Generación de casos de prueba a partir del perfil.
4. Ejecución de pruebas y registro y análisis de datos de fallas.
5. Cálculo y certificación de la fiabilidad.

La certificación para la ingeniería del software de sala limpia requiere la creación de 3 modelos:

* + ***Modelo de Muestreo:***La prueba del software ejecuta *m* casos de prueba aleatorios y se certifica si no ocurren fallas o un número específicos de estas.
  + ***Modelo de Componentes:***Se certificara un sistema compuesto de *n* componentes. El modelos de componentes permite que el analista determine la probabilidad de que el componente *i* fallara antes de completarse
  + ***Modelo de Certificación:***La fiabilidad global del sistema se proyecta y certifica.